E.U.I.T. TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA
PLAN 2000

TEMA:

TÍTULO: Desarrollo de una guía de aprendizaje para el manejo del entorno de CAD IspLEVER

AUTOR: Iván Romero González

TUTOR: Miguel Ángel Freire Rubio **Vº Bº.**

DEPARTAMENTO: SEC

Miembros del Tribunal Calificador:

PRESIDENTE: Sara Lana Serrano

VOCAL: Miguel Ángel Freire Rubio

VOCAL SECRETARIO: José Antonio Herrera Camacho

DIRECTOR: Miguel Ángel Freire Rubio

Fecha de lectura: 30/09/2013

Calificación: **El Secretario,**

RESUMEN DEL PROYECTO:

En este proyecto se analizan las características y el ciclo de diseño asociado al entorno de CAD IspLEVER, de Lattice Semiconductor, con la finalidad de evaluar su adecuación a la docencia relacionada con la ingeniería de sistemas digitales cableados. En base a este estudio se realiza una guía del manejo de las diferentes herramientas que se integran en el entorno.

Además, se realiza la caracterización de una serie de familias de dispositivos del fabricante Lattice Semiconductor que pudiera servir de apoyo a la hora de elegir un dispositivo de este fabricante para la realización de un determinado diseño.

Una vez realizada la caracterización de la familias, se expone el análisis de las mismas, enfocado a averiguar qué familias serían las más adecuadas para incluir alguno de sus miembros en una posible placa de prototipado utilizada a lo largo de un laboratorio semestral.

En este proyecto se analizan las características y el ciclo de diseño asociado al entorno de CAD IspLEVER, de Lattice Semiconductor, con la finalidad de evaluar su adecuación a la docencia relacionada con la ingeniería de sistemas digitales cableados. En base a este estudio se realiza una guía del manejo de las diferentes herramientas que se integran en el entorno.

Además, se realiza la caracterización de una serie de familias de dispositivos del fabricante Lattice Semiconductor que pudiera servir de apoyo a la hora de elegir un dispositivo de este fabricante para la realización de un determinado diseño.

Para dar comienzo a la realización del estudio del entorno y de las herramientas que integra IspLEVER, se procedió a la familiarización con el marco de trabajo. Esta familiarización se realizó, en un principio, a través de la lectura de la documentación ofrecida por el fabricante en su página web, http://www.latticesemi.com. Tras esta lectura, que sirvió para tener una primera visión de las características de la herramienta, se procedió a la descarga del paquete de instalación; el fabricante ofrece una versión de evaluación que expira a los 12 meses. Una vez descargado, se instaló y para terminar con los preparativos, se pasó el procedimiento de obtención de la licencia. Con ello se consiguió tener el software preparado para su utilización.

A continuación se emplearon horas de trabajo para, sin documentación alguna, tratar de crear diseños; con este trabajo se pretendía detectar lo intuitivo que resulta el entorno cuando se tienen conocimientos de herramientas de CAD electrónico.

Tras esta primera toma de contacto con el entorno real, se procedió al estudio de las diferentes opciones que ofrece para la realización de diseños, ya sean lógicos o físicos. Además del estudio de todas las posibilidades que ofrece el entorno, el trabajo se focalizó en la detección y comparación de las distintas opciones que ofrece para realizar una misma tarea, como ocurre con la asignación de pines o con la revisión de los resultados de una simulación, entre otras.

Entrelazado con el estudio de las opciones que ofrece el entorno, se realizó el estudio de las distintas herramientas de trabajo integradas en el mismo.

Una vez estudiado el entorno y las herramientas, se procedió a la realización del tutorial. Se capturaron todas las imágenes que se consideraron apropiadas para que al alumno le resultase cómodo y fácil seguir todas las indicaciones que el tutorial ofrece, para la realización de un ciclo de diseño lógico completo.

Tras la realización del tutorial, se procedió a revisar la amplia documentación que el fabricante ofrece de cada una de las distintas familias de dispositivos que fabrica. El fin de esta revisión no fue otro que realizar una caracterización de las distintas familias, que pudiera servir de apoyo a la hora de elegir un dispositivo de este fabricante para la realización de un determinado diseño. Este estudio de las familias de dispositivos del fabricante, también se realizó para detectar qué familia de dispositivos era la más idónea para incluir uno de sus miembros en una hipotética placa de prototipado, para la realización de prácticas de laboratorio.

This project consists in the analysis of the characteristics and the design cycle associated with the IspLEVER environment of CAD, by Lattice Semiconductor. The objective of that analysis is to evaluate their suitability for teaching engineering related to wired digital systems. Based on this analysis a guide was made for managing the different tools that are integrated into the environment.

In addition, the characterization of several families by the manufacturer Lattice Semiconductor was made, with the objective that it could be used to support the choice of a Lattice's device to perform a certain design.

To start the IspLEVER environment and tools study, I began with a familiarization with the environment. This familiarization consisted in a study of the manufacturer documentation offered in their web page, http://www.latticesemi.com. After that, I had a general view about the characteristics of the environment and environment tools. Then I continued downloading the installation package. The manufacturer offers an evaluation version that expires in the period of one year. After that download, the environment was installed. Finally, the licensing procedure was followed to finish with the preparations. Then, the software was prepared for its utilization.

Following, several work hours were wasted without documentation, trying to create designs. This work has been to identify how intuitive the environment is when you have knowledge of electronic CAD tools.

After this first point of contact with the real environment, I proceeded to study different offered options, by the manufacturer, for the realization of either logical or physical designs. In addition to studying all the

possibilities offered by the environment, the work is focused on the detection and comparison of the various options offered to perform the same task, as with the pin assignment or reviewing the results of a simulation…

At the same time, the environment tools were studied.

At this point, I began creating the tutorial. I captured all the figures that I consider important to make it easy to the students. The tutorial contains a complete logical design cycle.

When the tutorial was finished, I started to review the manufacturer documentation about each devices family. The purpose of this review was to characterize the different families to support the device selection in future designs. Another purpose of that characterization was focused on the detection of the best family to include one of its members in a prototyping board for conducting laboratory practices.

Índice:

I Capítulo I: “Introducción”

I.1 PROPÓSITO

Uno de los propósitos de este proyecto fin de carrera es la realización de un análisis del entorno y de las herramientas que integra IspLEVER. Este análisis se centra en la adecuación de las herramientas, que serían utilizadas por los alumnos, para familiarizarse con los entornos de CAD electrónico. Las condiciones en las que se desarrollaría la utilización del entorno por parte de los alumnos consistirían en un laboratorio tutorizado con el equipamiento básico presente en los laboratorios de electrónica de la escuela, fuente de alimentación, ordenadores con potencia de procesado suficiente y los programas ya instalados; además de conexión a Internet, necesaria para la utilización de la ayuda presente en el entorno.

Un segundo propósito es la preparación de un tutorial del manejo y del ciclo de diseño lógico, utilizando el entorno, que pueda servir como práctica inicial en un laboratorio para la enseñanza de los fundamentos de las herramientas de CAD. Con este tutorial se conseguirá una guía que los alumnos puedan seguir de forma autónoma; aunque en el laboratorio estaría presente un profesor, para poder resolver las dudas que se generasen. En esta guía, el alumno realizará un ciclo de diseño lógico completo, llegando hasta la verificación lógica del circuito a través de la realización de una simulación con retardos. Al obligar al alumno a seguir todos los pasos necesarios para la realización de un diseño, conseguiremos que el alumno comience a familiarizarse con todas las herramientas del entorno. Al mismo tiempo, se estarán afianzando en el alumno los conocimientos sobre el ciclo

de diseño, tanto en cuanto a las etapas necesarias, como en cuanto a su orden correcto.

El tercer propósito es la caracterización de una serie de familias de dispositivos del fabricante Lattice Semiconductors. Esta caracterización se ha enfocado a averiguar qué familias no serían aptas para incluir alguno de sus miembros en una posible placa de prototipado, cuales sí serían aptas y cual sería la más adecuada. En este análisis se ha tenido en cuenta si los dispositivos de cada familia presentan funcionalidades que no serían utilizadas a lo largo de un laboratorio semestral, que parte de un nivel básico como es el del tutorial que será realizado. También se ha tenido en cuenta si los dispositivos de cada familia son o no capaces de materializar los posibles diseños de mediana complejidad que podrían ser exigidos. A parte de este enfoque, tan centrado en la decisión de qué familia utilizar para la realización de placas de prototipado para el laboratorio, también se describirán las características de cada familia objeto de estudio, de tal modo que pueda servir como guía para facilitar la elección de la familia a utilizar en futuros diseños.

I.2 DESCRIPCIÓN DEL TRABAJO REALIZADO

Con el fin de alcanzar cada uno de los propósitos planteados en el apartado anterior, se han realizado los trabajos que se detallan a continuación.

Para dar comienzo a la realización del estudio del entorno y de las herramientas que integra IspLEVER, se procedió a la familiarización con el marco de trabajo. Esta familiarización se realizó, en un principio, a través de la lectura de la documentación ofrecida por el fabricante en su página web, http://www.latticesemi.com. Tras esta lectura, que sirvió para tener una primera visión de las características de la herramienta, se procedió a la descarga del paquete de instalación; el fabricante ofrece una versión de evaluación que expira a los 12 meses. Una vez descargado, se instaló y para terminar con los preparativos, se pasó el procedimiento de obtención de la licencia. Con ello se consiguió tener el software preparado para su utilización.

A continuación se emplearon horas de trabajo para, sin documentación alguna, tratar de crear diseños; con este trabajo se pretendía detectar lo intuitivo que resulta el entorno cuando se tienen conocimientos de herramientas de CAD electrónico.

Tras esta primera toma de contacto con el entorno real, se procedió al estudio de las diferentes opciones que ofrece para la realización de diseños, ya sean lógicos o físicos. Además del estudio de todas las posibilidades que ofrece el entorno, el trabajo se focalizó en la detección y comparación de las distintas opciones que ofrece para realizar una misma tarea, como

ocurre con la asignación de pines o con la revisión de los resultados de una simulación, entre otras.

Entrelazado con el estudio de las opciones que ofrece el entorno, se realizó el estudio de las distintas herramientas de trabajo integradas en el mismo.

Una vez estudiado el entorno y las herramientas, se procedió a la realización del tutorial. Se capturaron todas las imágenes que se consideraron apropiadas para que al alumno le resultase cómodo y fácil seguir todas las indicaciones que el tutorial ofrece, para la realización de un ciclo de diseño lógico completo. Se eligió un circuito básico para que el aprendizaje se centrara en el entorno y no en las características que el circuito puediera ofrecer.

Tras la realización del tutorial, se procedió a revisar la amplia documentación que el fabricante ofrece de cada una de las distintas familias de dispositivos que fabrica. El fin de esta revisión no fue otro que realizar una caracterización de las distintas familias, que pudiera servir de apoyo a la hora de elegir un dispositivo de este fabricante para la realización de un determinado diseño. Este estudio de las familias de dispositivos del fabricante, también se realizó para detectar qué familia de dispositivos era la más idónea para incluir uno de sus miembros en una hipotética placa de prototipado, para la realización de prácticas de laboratorio.

I.3 DESCRIPCIÓN DE LA ESTRUCTURA

El proyecto fin de carrera se ha estructurado en 5 capítulos, de la siguiente manera:

El primer capítulo (Capítulo I: “Introducción”) recoge la introducción de este proyecto fin de carrera y es donde se comentan las bases de lo que se quiere conseguir con este trabajo, en qué consiste el trabajo realizado, cómo se organiza el presente documento y para terminar, qué documentación se utilizó para la realización del mismo, así como dónde encontrar dicha documentación.

El segundo capítulo (Capítulo II: “El entorno de CAD ispLEVER Classic 1.2”) contiene el tutorial para la realización de un ciclo de diseño lógico completo, incluyendo la captura de esquemas y simulación lógica, la implementación del diseño y por último la verificación del diseño.

El tercer capítulo (Capítulo III: “Tecnologías de FPGAs y PLDs del fabricante Lattice”) presenta, por separado, las características de cada una de las familias de dispositivos del fabricante. En primera instancia se divide en FPGAs y PLDS y en un segundo nivel se alcanza la división por familias. En el último apartado de este capítulo se expresan las conclusiones sobre qué familia de dispositivos sería la más adecuada para incluir uno de sus miembros en la placa de prototipado para la realización de un laboratorio.

El cuarto capítulo (Capítulo IV: “Conclusiones”) contiene las conclusiones respecto a la viabilidad del entorno para la enseñanza de herramientas de CAD.

El último capítulo (Capítulo V: “Documentación del fabricante”) contiene imágenes extraídas de las hojas de características, publicadas por el fabricante, donde se exponen las principales bonanzas de cada familia de dispositivos estudiada.

I.4 DOCUMENTACIÓN

La totalidad de la documentación utilizada, para la realización de este proyecto fin de carrera, se ha extraído de la página web del fabricante Lattice Semiconductors (http://www.latticesemi.com); en la cual, para encontrar la documentación deseada, hay una herramienta de búsqueda que resulta muy útil por el gran volumen de información que ofrece el fabricante.

II Capítulo II: “El entorno de CAD ispLEVER Classic 1.2”

II.1 INTRODUCCIÓN

El software de CAD ispLEVER ha sido desarrollado por Lattice para la realización de diseños de circuitos digitales, utilizando las familias de dispositivos que comercializa. Se trata de un entorno profesional y fácil de manejar, con un gran nivel de integración de las herramientas que lo componen. Además, Lattice proporciona versiones de uso libre durante un tiempo suficientemente largo; la licencia gratuita dura un año para aplicaciones profesionales y docentes.

Utilizando el software de CAD ispLEVER se pueden realizar ciclos completos de diseño de circuitos digitales. Estos ciclos pueden ser lógicos, los cuales culminan con la verificación del diseño realizado; o físicos, que son aquellos que culminan con la programación del circuito digital en un dispositivo físico, de ahí su nombre.

En líneas generales, los ciclos de diseño lógicos constan de dos fases principales: la especificación del diseño y la simulación lógica. Para la especificación de los diseños, el entorno presenta una herramienta para la captura de esquemas y editores, que permiten especificar los diseños a través de Lenguajes de Descripción Hardware (VHDL y Verilog). El entorno permite que en un mismo diseño se utilicen ambos métodos, lo cual favorece la reutilización de módulos en los diseños.

En la primera de las fases, la captura de esquemas, el entorno ispLEVER permite realizar tanto esquemáticos planos como esquemáticos jerárquicos, en los que los diseños están definidos en más de un nivel. El editor de esquemáticos se combina con el navegador jerárquico, el editor de símbolos y las librerías de símbolos para permitir la revisión y la modificación de los diseños de una manera cómoda para el usuario. Ya que todas ellas son herramientas lo suficientemente intuitivas, no se necesita más que una única utilización guiada para conseguir entender su funcionamiento. Y de este modo ser capaz de utilizarlas cuando sea necesario, sin necesidad de revisar documentación alguna. Dentro de esta primera fase, el entorno realiza la comprobación de las reglas de diseño. Esta comprobación se encarga de alertar al usuario de las posibles violaciones que presente el esquemático, como podría ser un bucle cerrado o un cortocircuito, por ejemplo.

En la segunda, la simulación lógica, el entorno ofrece la posibilidad de realizar una simulación funcional. Esta simulación funcional es un proceso que permite la detección de errores en la realización del diseño lógico. En una simulación funcional no se tienen en cuenta los tiempos de propagación. Estas simulaciones funcionales sirven para saber, si en este punto del diseño, el circuito se comportará de la forma deseada. Para realizar estas simulaciones es necesario definir las formas de onda de las señales que son entradas en el circuito diseñado. El entorno ofrece varias formas de definir las formas de onda usadas en las simulaciones. Es posible importar archivos que contengan los vectores de test, definir los estímulos de forma textual o definir los estímulos de forma gráfica a través del editor de formas de onda que incorpora la herramienta. Aunque sí es cierto que se aprecia el interés del fabricante por intentar facilitar la labor de definición de formas de onda, al posibilitar la definición a través de 3 métodos

distintos; el resultado no es del todo satisfactorio ya que la definición de formas de onda, sobre todo si han de tener saltos a intervalos no regulares, sigue siendo una labor lenta y que requiere del usuario la realización de muchos pasos hasta conseguir la forma deseada.

Estas dos fases se han de ejecutar de forma secuencial y reiterativa, hasta conseguir que el resultado de la simulación lógica completa del diseño sea el esperado. En este punto nos volvemos a encontrar con deficiencias en el entorno, debido a las pocas facilidades que ofrece la herramienta de visualización de formas de onda.

Con la realización de una simulación lógica completa del diseño, en la que el comportamiento de las salidas de nuestro circuito sea el esperado, termina el ciclo de diseño lógico.

Los ciclos de diseño físicos constan, además de las fases que componen un diseño lógico, de otras tres fases que son: la implementación del diseño, la verificación del mismo y la programación del dispositivo.

La implementación del diseño consiste en ajustar el diseño a un dispositivo concreto, en el entorno la implementación del diseño incluye: la definición de las restricciones deseadas por el usuario, la compilación, la optimización y la adecuación del diseño. En el entorno se ejecutan de forma automática y secuencial la compilación, la optimización y la adecuación del diseño en un solo acto; con lo que el usuario solo tendrá que definir, si las tiene, las restricciones específicas que quiera que cumpla el diseño cuando sea programado en un dispositivo físico.
La verificación del diseño consiste en comprobar que el comportamiento real del diseño cubrirá las necesidades que dieron origen a las

especificaciones del diseño, para ello el entorno ofrece dos posibilidades. La primera opción es el análisis estático de tiempos, con el cual podremos comprobar los parámetros temporales estáticos del diseño, como son, el camino crítico, los tiempos de set-up y hold y la máxima frecuencia de reloj entre otros. La segunda opción es la simulación con retardos, a cuya salida se presentará la forma de onda de las señales de nuestro diseño; estas formas de onda podrán ser revisadas y medidas utilizando el editor de formas de onda, lo que nos permitirá comprobar si nuestro diseño cumplirá con las especificaciones cuando sea programado sobre un dispositivo real.

Estas dos fases también han de ejecutarse de forma secuencial y reiterativa hasta conseguir que el diseño cumpla con las especificaciones. Es bastante habitual que un diseño, que no pueda cumplir con ciertas especificaciones al ajustarlo a un determinado dispositivo, sí que cumpla, si es ajustado a un dispositivo con mayor capacidad o mejores características. Estas situaciones son las que obligan a la reiteración de estas dos fases del diseño físico.

Una vez que se tiene el diseño verificado, se puede pasar a la última fase del diseño físico; que consiste en la programación del dispositivo real y las pruebas del mismo para comprobar que se ajusta a las especificaciones. El entorno incluye, para la programación del dispositivo real, una herramienta denominada ispVM System.

II.1.1 Descripción del alcance del tutorial

Este tutorial pretende servir de primera práctica guiada a los alumnos, que se enfrenten a su primera experiencia, ante un entorno de CAD electrónico.

Para ello, se incluye un primer apartado en el que se indica al alumno dónde encontrar el software y los requisitos de éste. Posteriormente se explica el uso de algunos de los tipos de archivos utilizados en el entorno. También se detallan los pasos necesarios para la creación de un nuevo proyecto y la selección de un dispositivo. El entorno obliga a la realización de estos dos últimos pasos antes del comienzo del ciclo de diseño lógico descrito en la introducción de este capítulo.

Para realizar el ciclo de diseño lógico completo, este tutorial guiará al alumno desde la especificación del diseño hasta la verificación del mismo tras una simulación con retardos.

En la fase de especificación del diseño, la posibilidad elegida ha sido la captura de esquemas. Esta decisión fue tomada en base a que para poder entender una especificación del diseño a través de los editores, que permiten utilizar Lenguajes de Descripción Hardware, el alumno tendría que tener conocimientos en estos lenguajes; y estos conocimientos podría no poseerlos en el momento de utilizar esta guía. Las posibilidades referentes a la especificación de diseño, explicadas en este tutorial, incluyen: cómo añadir un esquemático al proyecto, cómo reajustar el tamaño de la hoja de un esquemático, cómo emplazar símbolos desde una librería de símbolos, cómo utilizar el zoom, cómo realizar la interconexión de instancias, cómo etiquetar los nodos del diseño, cómo añadir marcadores

de entradas y salidas y cómo crear un símbolo. Tras estas explicaciones el alumno tendrá un circuito capturado en su proyecto y conocimientos para poder capturar todos aquellos circuitos que pueda necesitar.

Seguidamente, se guiará al alumno para que realice la comprobación de las reglas de diseño y para que edite las formas de onda necesarias para la realización de una simulación funcional. Esta simulación será realizada a continuación. Para terminar esta fase del ciclo de diseño lógico se realizará la comprobación de los resultados de la simulación funcional.

Tras la especificación del diseño, se realizará la implementación del mismo, en la que se incluyen las siguientes partes: asignación de pines, materialización del diseño (adaptación del diseño al dispositivo elegido), configuración de las opciones de visualización de los informes y lectura del informe de adaptación.

Por último, se realizará la verificación del diseño a través del análisis estático de tiempos y la simulación con retardos. Esta simulación con retardos será revisada utilizando el visor de formas de onda. Para terminar, se muestra como visualizar el estado de las señales del circuito en un instante concreto de una simulación, gracias a la correlación entre los resultados de una simulación y el esquemático.

II.1.2 Explicación de la estructura del tutorial

Este tutorial se estructura en 4 grandes apartados. Tras un apartado inicial en el que se expone la introducción a este capítulo, se presentan otros tres apartados que son: captura de esquemas y simulación lógica, implementación del diseño y verificación del diseño. Estos apartados están divididos en distintos subapartados donde se guía al alumno para que realice las distintas acciones necesarias para completar un ciclo de diseño lógico.

II.2 CAPTURA DE ESQUEMAS Y SIMULACIÓN LÓGICA

II.2.1 Introducción

Para aprender a realizar simulaciones lógicas, y a manejar las distintas herramientas implicadas en el proceso, se va a realizar, en primer lugar, un ejemplo guiado con un circuito muy sencillo. En la figura puede observarse el esquema del circuito, que sirve para acondicionar la duración de un pulso de entrada a un circuito.

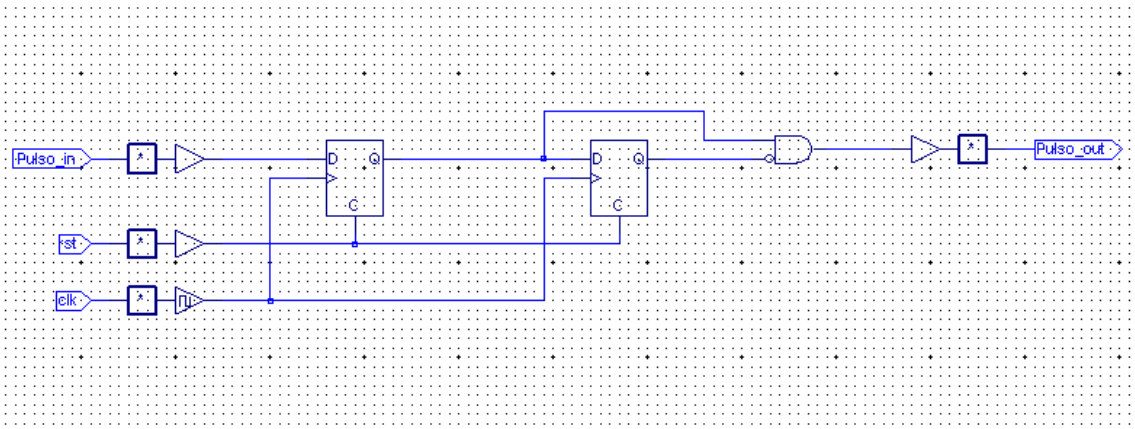


Figura II.1

En el desarrollo guiado del ejemplo se utilizarán las convenciones que se señalan a continuación.

1.- Las acciones dirigidas que deben ser realizadas para completar el ejercicio van precedidas del texto “siga el siguiente procedimiento.” Antes de ejecutarlas, lea atentamente todos los pasos que describen el procedimiento completo.

2.- Cuando el desarrollo de una acción requiera la utilización del teclado del ordenador y sea preciso pulsar más de una tecla se indicará de la siguiente manera: **Tecla1 + Tecla2**.

II.2.1.1 Instalación del software

En esta guía se va a tomar como referencia la versión Classic 1.2, de libre distribución para su prueba durante un periodo de un año, del entorno ispLEVER de Lattice. Los ejecutables para la instalación de dicho entorno pueden obtenerse directamente del sitio de Internet del fabricante www.latticesemi.com, donde se encuentran instrucciones precisas sobre los pasos a seguir para una correcta instalación del software y también para la obtención de la licencia. Para la obtención de la licencia será necesario conocer la dirección MAC del equipo donde vayamos a instalar el software y además hemos de indicar una dirección de correo electrónico, donde el fabricante enviará el fichero de licencia.

Los requisitos hardware para la correcta instalación del entorno son los siguientes:

Intel Pentium o PC compatible con Pentium.
Windows 7, Windows XP, Windows 2000 Workstation o 32-bit Windows Vista.
Mínimo 512 MB de memoria, recomendado 1 GB.
Aproximadamente 3 GB de espacio libre en disco.

II.2.1.2 Iniciando la aplicación ispLEVER Classic 1.2

Siga el siguiente procedimiento.

1.-Inicie la aplicación **ispLEVER Classic Project Navigator** en **Windows**.

Esta acción abrirá el navegador de proyectos de la herramienta. Éste se encarga de que los ficheros, generados como salida y necesarios como entrada para las distintas herramientas, se comuniquen con corrección entre las mismas. Su aspecto es el mostrado en la figura.

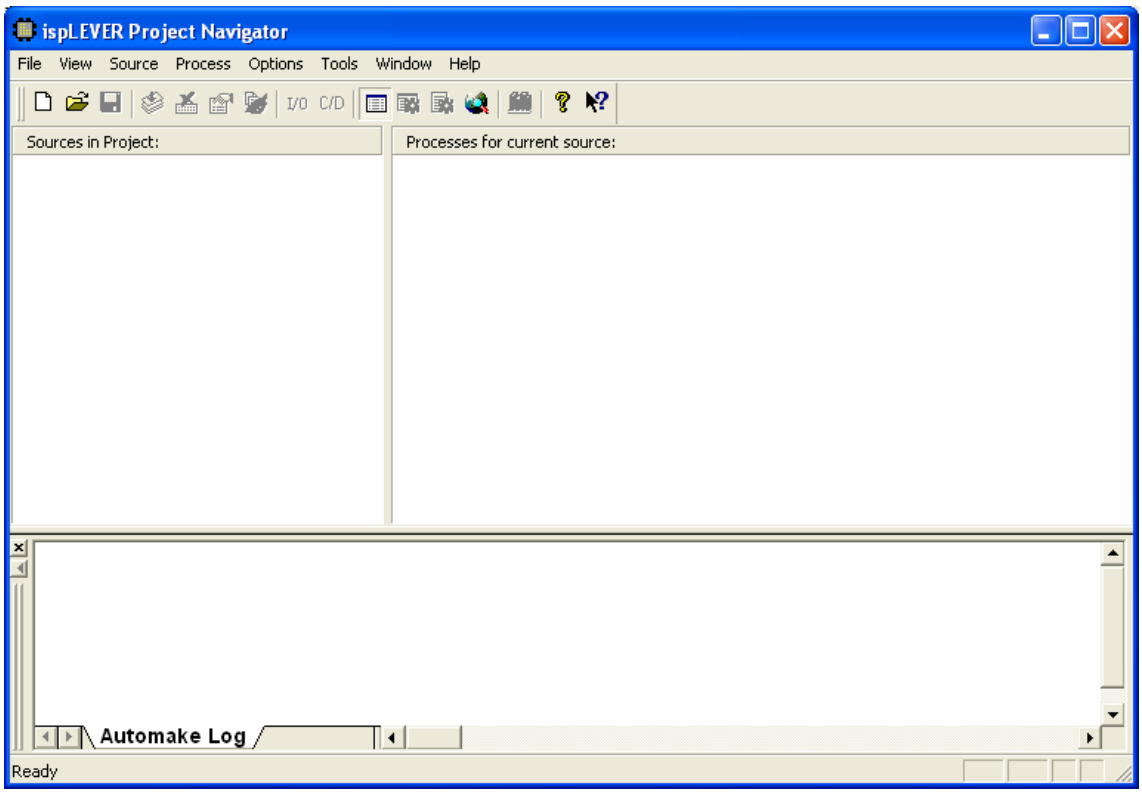


Figura II.2

II.2.1.3 Finalización de una sesión de trabajo

Esta acción se realiza exactamente igual que en cualquier aplicación montada sobre Windows, se puede salir mediante un clic en el aspa de la esquina superior derecha, o bien, dentro del menú **File** del navegador de proyectos hay una opción llamada **Exit** que cierra el mismo.

II.2.2 La Herramienta de Ayuda

Con anterioridad a la realización de este tutorial conviene conocer la existencia de la ayuda que, a través de una conexión a Internet, proporciona gran cantidad de información sobre cómo realizar acciones que pueden ser necesarias para los diseños que estemos implementando.

II.2.3 Proyectos en el entorno ispLEVER Classic 1.2

Lo primero que hay que realizar para comenzar a trabajar en nuestro diseño, es definir el proyecto de trabajo sobre el cual vamos a trabajar. Pero en este tutorial antes de ello vamos a exponer ciertas nociones sobre algunos tipos de archivos presentes en los proyectos.

II.2.4 Ficheros de diseño y ficheros auxiliares

Un proyecto está constituido por un conjunto de ficheros almacenados en un directorio del sistema operativo. Los ficheros pueden clasificarse en dos categorías: **ficheros de diseño** y **ficheros auxiliares**.

Un fichero de diseño es un fichero gráfico o de texto, creado con un editor del entorno **ispLEVER Classic 1.2** o, también, con otro esquemático estándar, editor de texto o generador de **netlist** en formato EDIF, Verilog o VHDL. Estos ficheros contienen la especificación lógica de los diseños que pueden ser procesados por el compilador para generar modelos de simulación y/o ficheros de programación de dispositivos (ficheros que sirven para que un chip fabricado por Lattice implemente el circuito que se ha diseñado). El compilador puede procesar automáticamente distintos tipos de ficheros, que se identifican por una determinada extensión. En este tutorial se van a utilizar ficheros realizados con la captura de esquemas del entorno. Su extensión es **.abl** (**ABEL-HDL** que es el capturador de esquemas del entorno) o **.sch** (**schematic**).

Los ficheros auxiliares son los que están asociados a un proyecto **ispLEVER Classic 1.2** sin formar parte de la jerarquía de diseño, pudiendo ser, en algunos casos, editables por el usuario; entre ellos se encuentran todos los ficheros de salida de las aplicaciones del entorno (exceptuando, por supuesto, los ficheros de diseño). A continuación se presentan algunos de los que resultan interesantes para la captura y simulación de circuitos:

1.- Ficheros de símbolos: Son ficheros que contienen símbolos de los esquemas del diseño; se utilizan para la realización de diseños jerárquicos

(diseños en los que se incrustan módulos diseñados por el usuario), la extensión de estos ficheros es .**sym**.

2.- Ficheros de simulación: Contienen la información de las señales presentes en la simulación. Son generados por el usuario en formato texto (**.abv/.abl**), o formas de onda generadas a través del editor de formas de onda (.**wdl**). Son utilizados por el simulador lógico para la ejecución de simulaciones.

Para indicar al entorno el proyecto de trabajo sobre el cual queremos actuar hay que indicarle el nombre del archivo de más alto nivel (**.syn**) y el directorio donde se encuentre ubicado, más adelante el navegador de proyectos usará este archivo para retomar el proyecto.

II.2.5 Creación de un nuevo proyecto

Para la creación de un nuevo proyecto, siga el siguiente procedimiento.

1.- Arranque el entorno **ispLEVER**, si no está en ejecución.

2.- En el navegador de proyectos seleccione **File** > **New Proyect** para abrir la ventana de diálogo que permite crear proyectos. Esta ventana de diálogo se llama **Proyect Wizard**.

3.- En dicha ventana de diálogo rellene los campos como se indica a continuación:

a.- En el apartado **Proyect Name** teclee Pulso_IN_Adapter.

b.- En el apartado **Location** teclee el siguiente directorio:

<install_path>\<Grupo_Lab>\Tutorial

c.- En el apartado **Design Entry Type** seleccione **schematic/ABEL**.

d.- En el apartado **Synthesis Tool** seleccione **Synplify**.

e.- Haga clic en el botón **Siguiente >**.

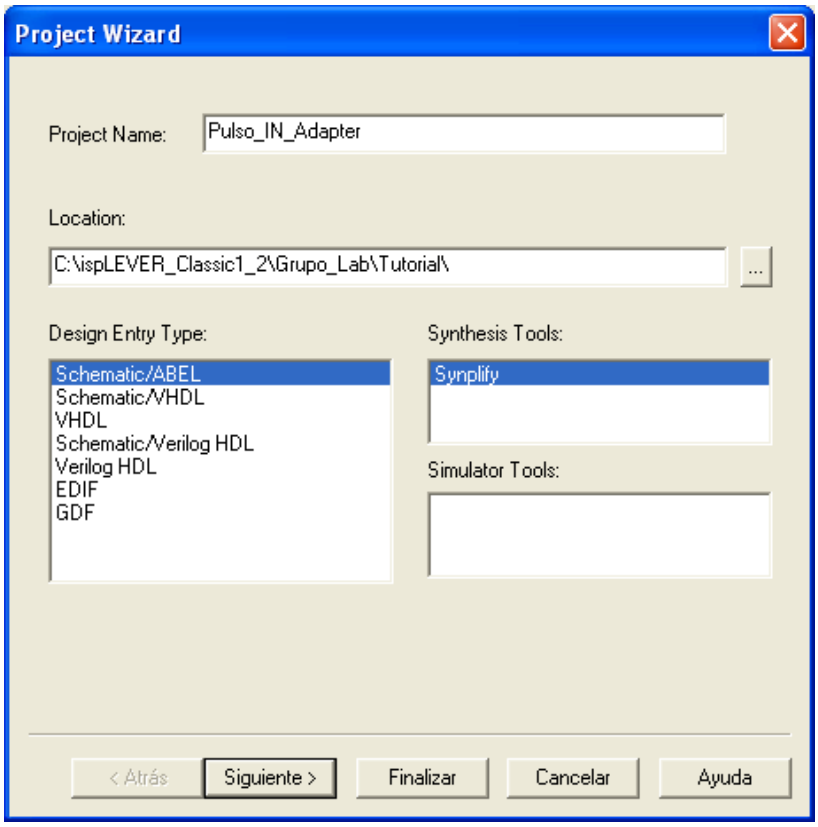


Figura II.3

II.2.6 Seleccionar un dispositivo

En la ventana fuente del **Proyect Navigator** es el texto a la derecha del icono  el que permite acceder a la ventana **Device Selector** tras hacer doble clic, esta ventana permite cambiar el dispositivo seleccionado en cualquier momento del proceso de diseño.

Para la selección del dispositivo físico en el que implementaremos nuestro diseño.

Tras la pulsación del botón **Siguiente** indicado en el apartado anterior aparecerá la ventana de diálogo **Proyect Wizard – Select Device**.

Para la selección del dispositivo sobre el que irá montado nuestro diseño, siga el siguiente procedimiento.

1.- En la ventana de diálogo **Proyect Wizard – Select Device** haga lo siguiente:

a.- En el apartado **Family** seleccione **ispMACH 4000**.

b.- En el apartado **Device** seleccione **LC4256V.**

c.- Acepte la configuración por defecto y pulse el botón **Siguiente** >.

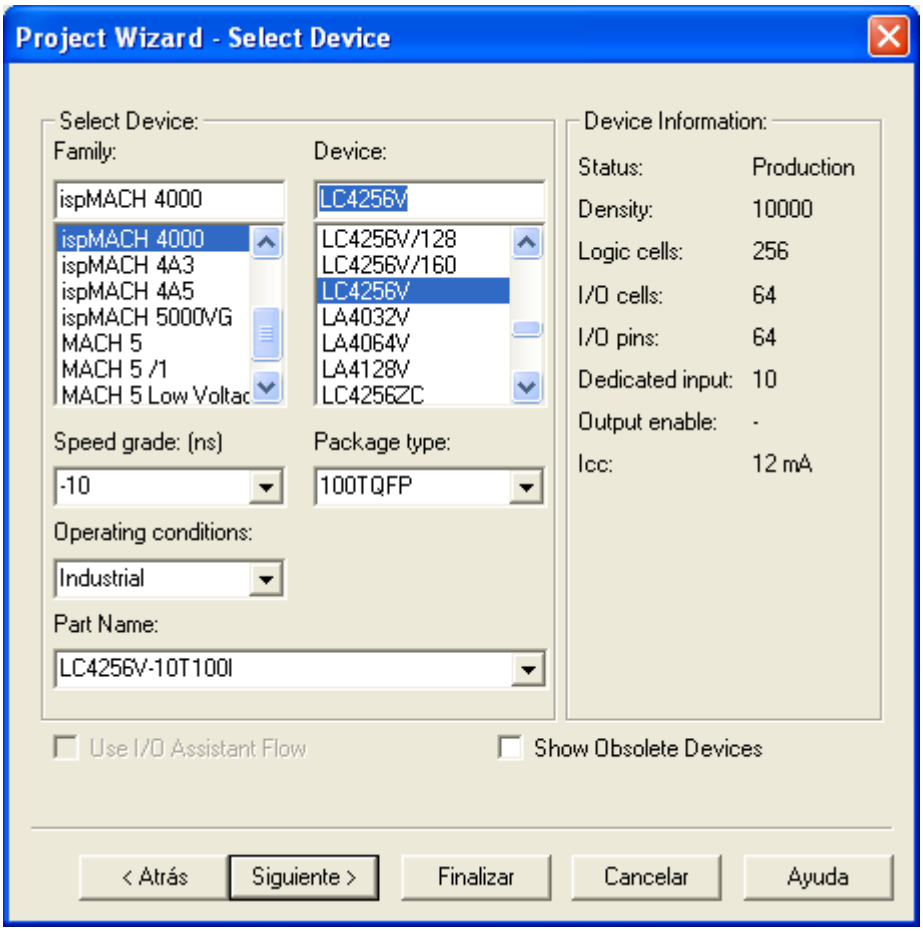


Figura II.4

d.- Pulse el botón Siguiente para aceptar el cuadro de diálogo **Add Source**.

e.- Pulse el botón **Finalizar** en el cuadro de diálogo **Proyect Information**.

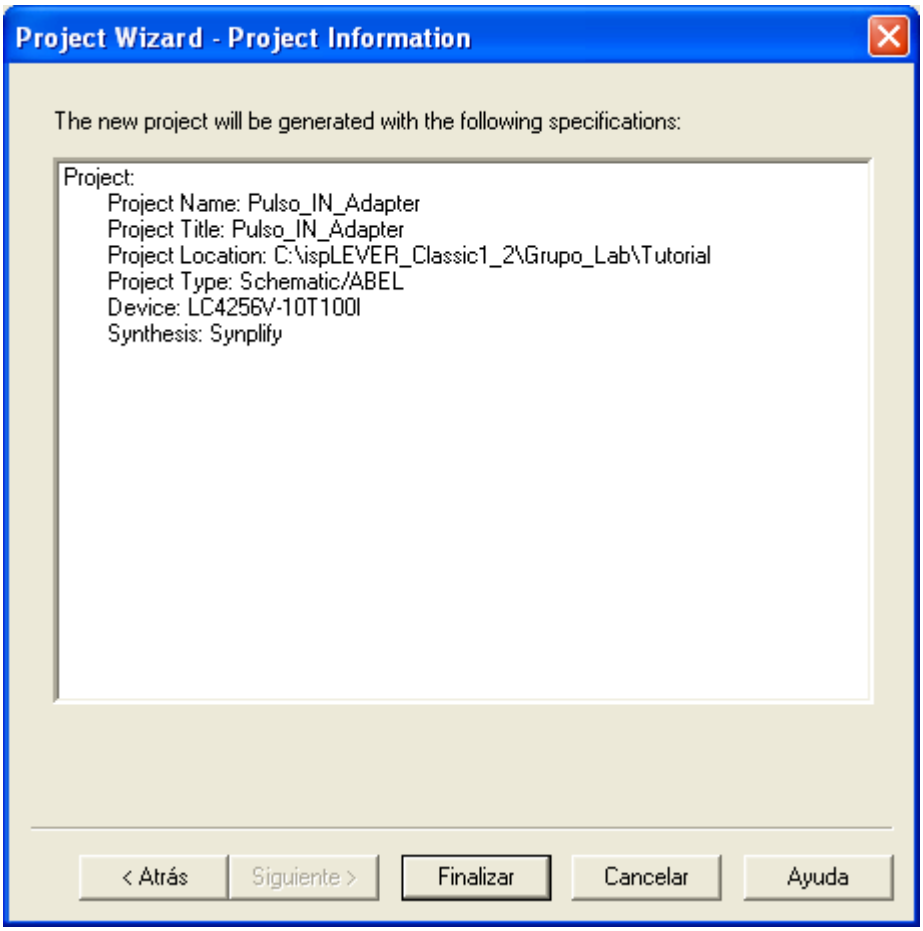


Figura II.5

II.2.7 Especificación del diseño

Una vez llegados a este punto en el cual hemos definido el proyecto de trabajo y el dispositivo que materializará nuestro diseño, ha de definirse el diseño lógico del circuito que se pretende realizar.

El entorno ispLEVER permite realizar tanto esquemáticos planos (de un único nivel jerárquico) como esquemáticos jerárquicos (de más de un nivel). Los diseños pueden definirse en esquemáticos de múltiples hojas. El editor de esquemáticos se combina con el navegador jerárquico (Hierachy Navigator), el editor de símbolos y las librerías de símbolos para permitir la revisión y la modificación de los diseños de una manera cómoda para el usuario.

A la hora de definir un esquemático disponemos de dos opciones. La primera de ellas consiste en crear los archivos del diseño paso a paso mediante las herramientas que integra el entorno. Y la segunda opción consiste en importar archivos de diseño ya realizados.

En nuestro caso nos decantaremos por la primera opción utilizando la captura de esquemas para definir el diseño de nuestro circuito.

II.2.7.1 Añadir un esquemático al proyecto

Para añadir un nuevo esquemático al proyecto, siga el siguiente procedimiento.

1.- En el navegador de proyectos seleccione **Source** > **New** para abrir la ventana de diálogo **New Source.**

2.- Seleccione **Schematic** y pulse el botón **OK**.

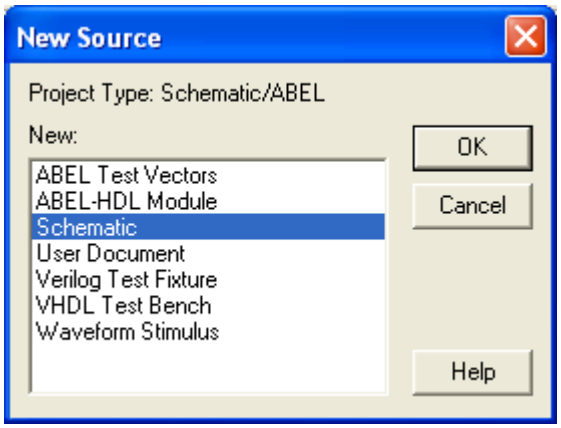


Figura II.6

Se abrirá una ventana en la que hemos de indicar el nombre del nuevo fichero esquemático.

3.- Escriba Pulso_IN_Adapter y presione el botón **OK**.

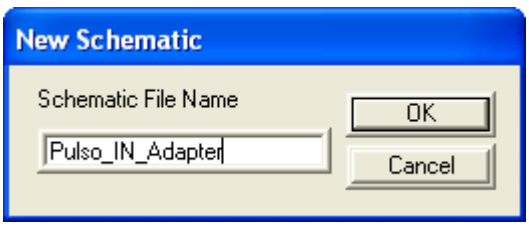


Figura II.7

Tras esta acción se abrirá el editor de esquemáticos y éste llamará a la hoja actual PULSO_IN_ADAPTER, además el entorno añadirá el nuevo archivo fuente pulso_in_adapter.sch al proyecto actual.

II.2.7.2 Reajustar el tamaño de la hoja de un esquemático

El editor de esquemáticos permite reajustar el tamaño de la hoja en cualquier momento.

Para ajustar el tamaño de una hoja de esquemático, siga el siguiente procedimiento.

1.- En el editor de esquemáticos seleccione **File** > **Sheets**, esto nos permite seleccionar la hoja sobre la que queremos actuar, en nuestro caso solo tenemos una.

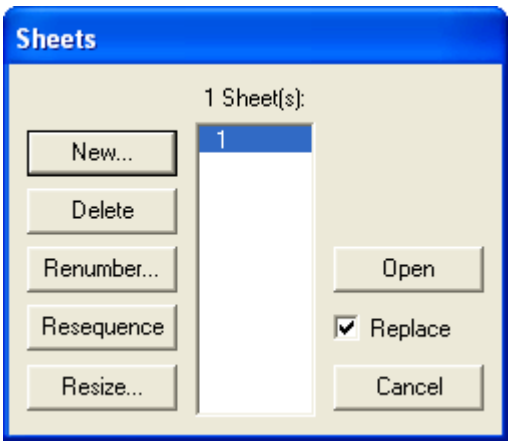


Figura II.8

2.- Pulse sobre el botón **Resize** y se abrirá la ventana de diálogo **Resize Sheet** en la cual se muestra un listado con los posibles tamaños de hoja, el tamaño actual se presenta sombreado.

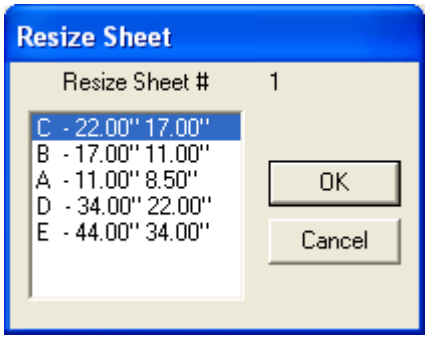


Figura II.9

3.- Seleccione el tamaño **B** y pulse sobre el botón **OK** para cerrar la ventana de diálogo **Resize Sheet**.

4.- De nuevo en la ventana de diálogo **Sheets** pulse sobre el botón **Open** para que el programa reajuste el tamaño de la hoja especificada.

II.2.7.3 Emplazar símbolos desde una librería de símbolos

El dispositivo que se ha seleccionado para implementar nuestro diseño determina las librerías de símbolos que estarán disponibles en el editor de esquemas. Si utilizamos símbolos de la librería de símbolos genéricos podremos cambiar el dispositivo en el cual implementaremos nuestro diseño sin tener que volver a dibujar un nuevo esquemático.

Para emplazar símbolos de la librería de símbolos genéricos llamada “regs.lib”, siga el siguiente procedimiento.

1.- En el editor de esquemáticos seleccione **Add > Symbol** para abrir la ventana de diálogo **Symbol Libraries**; esta acción se puede abreviar pulsando la tecla **F2**.

2.- En la ventana de diálogo **Symbol Libraries**, en el cuadro **Library** seleccione generic\regs.lib.

3.- En el cuadro **Symbol** seleccione **G_DC**.

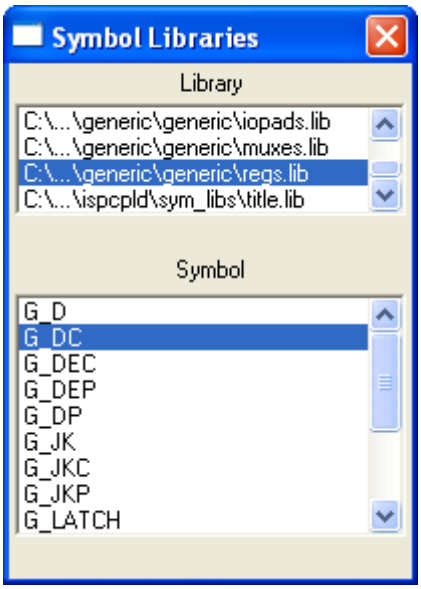


Figura II.10

El símbolo de un flip-flop tipo D se adjunta al cursor del ratón y haciendo clic sobre el esquemático emplazamos tantos flip-flop como deseemos, en nuestro caso dos. Haciendo clic con el botón derecho se dejan de insertar más unidades.

4.- Sobre el esquemático, haga dos veces clic para que el esquemático quede como en la figura que se muestra a continuación.

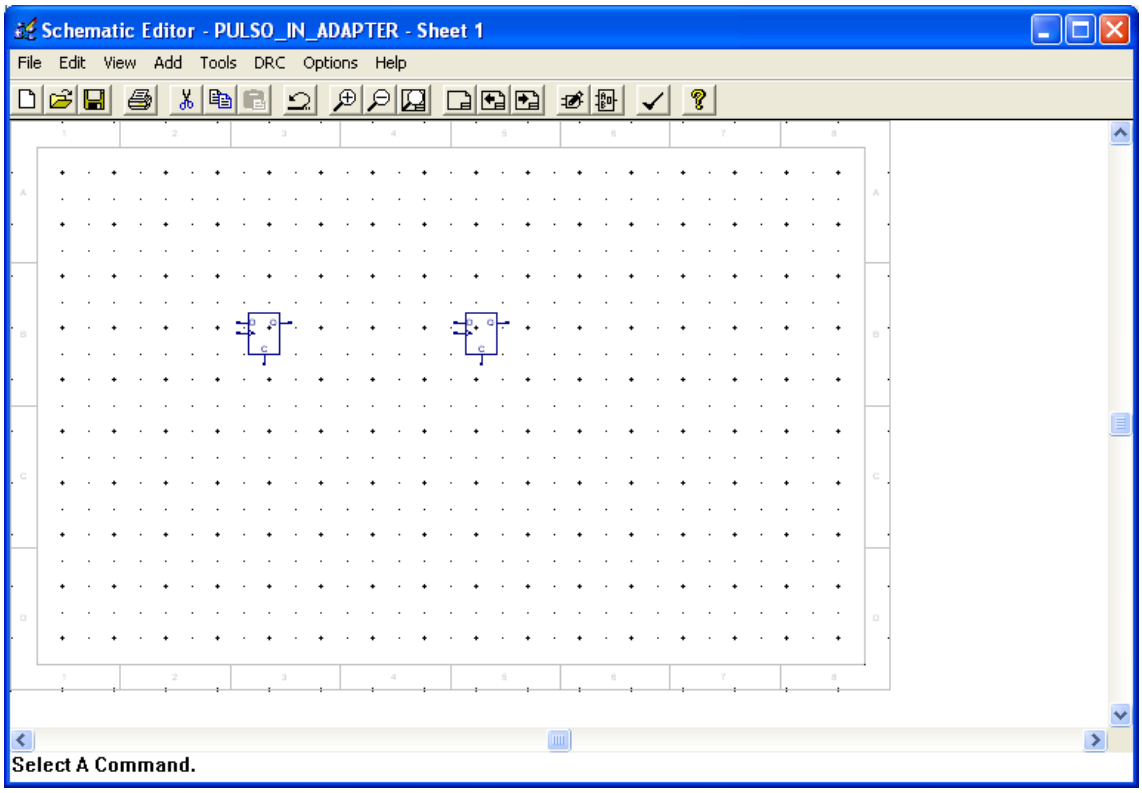


Figura II.11

5.- Con el mismo procedimiento emplace un símbolo **G_2AND1** de la librería **gates.lib** y de la librería **iopads.lib** emplace un símbolo **G_CLKBUF**, otro **G_INPUT** y otro **G_OUTPUT**, quedando el esquemático como indica la figura II.12.

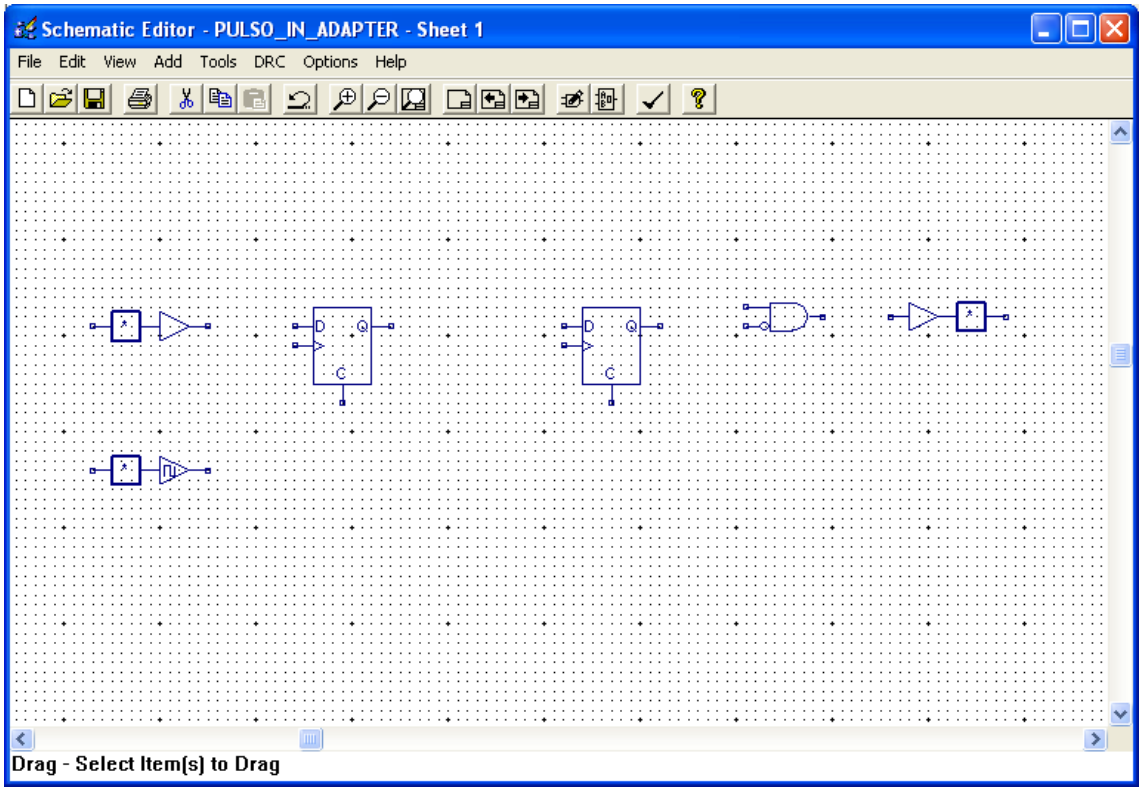


Figura II.12

II.2.7.4 Uso del zoom

Para poder dibujar esquemáticos con mayor comodidad el editor dispone de comandos de zoom que pueden ejecutarse por medio de los botones  .

II.2.7.5 Interconexión de instancias

Para interconectar puntos del esquemático hemos de usar el elemento llamado **wire** que sirve para representar conexiones entre símbolos.

Para interconectar los símbolos del esquemático, siga el siguiente procedimiento.

1.- Seleccione **Add > Wire**.

2.- Pulse en el extremo de cada conexión para que el esquemático quede como en la figura II.13.

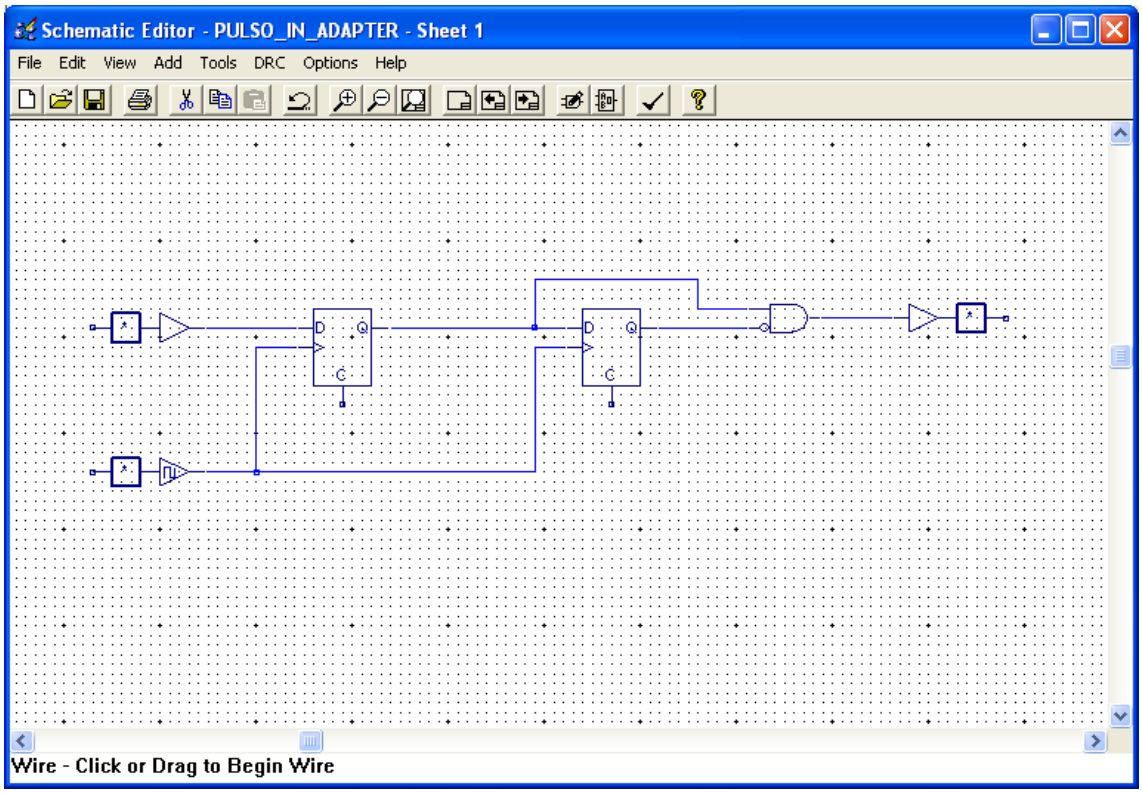


Figura II.13

3.- Emplace otro símbolo **G_INPUT** y conéctelo para que el esquema quede como se indica a continuación.

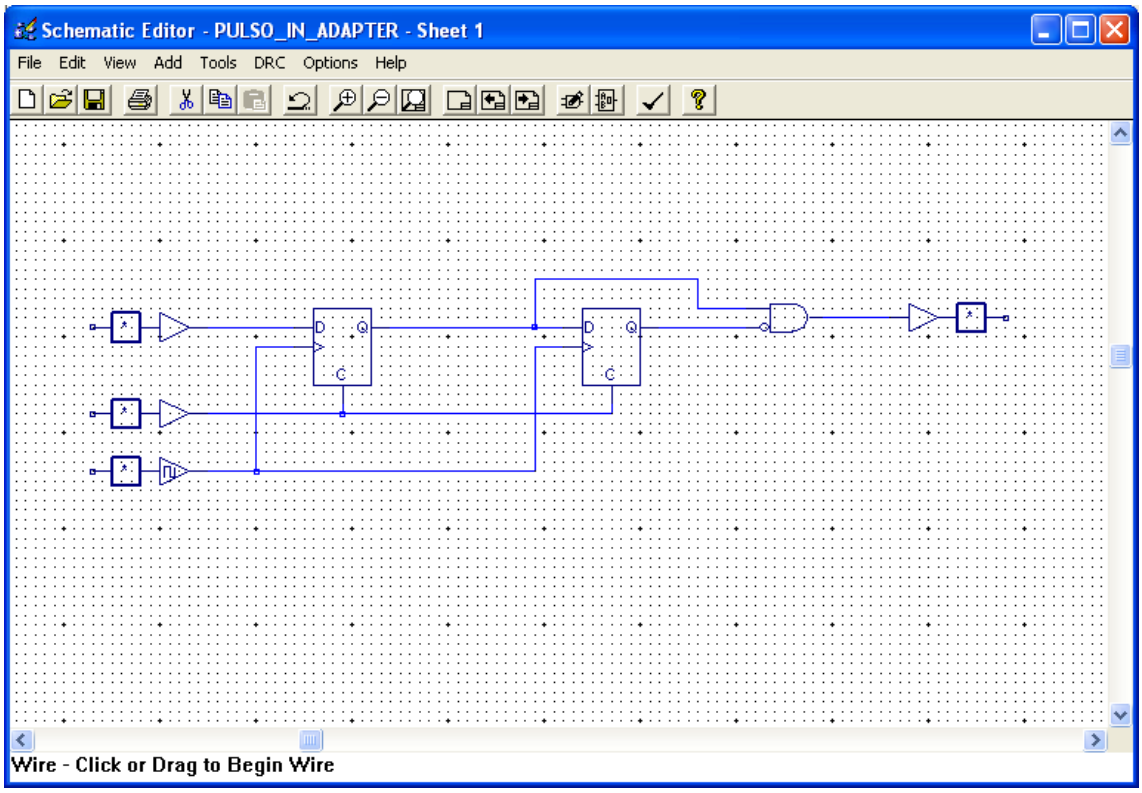


Figura II.14

II.2.7.6 Etiquetado de los nodos del diseño

Todos los nodos del diseño tienen un nombre, asignado por nosotros o, en su defecto, por el editor de esquemáticos.

Se pueden etiquetar los nodos de uno en uno y también se puede crear un nombre compuesto que se irá disgregando secuencialmente en los nombres individuales a asignar en los distintos nodos del esquema.

Para añadir varios nombres a distintos nodos del esquema de una sola vez, siga el siguiente procedimiento.

1.- Seleccione **Add > Net Name.**

2.- Escriba, en la línea de comando (la línea inferior de la ventana del editor de esquemáticos) **clk,rst,Pulso_in,Pulso_out** y pulse **Enter**, la coma separa nombres simples.

3.- Todo el nombre se adjunta al cursor del ratón, para separarlo en sus partes hay que hacer clic con el botón derecho del ratón y entonces haciendo clic en los distintos nodos vamos dándoles nombre.

Nota: si no permite etiquetar algunas entradas o salidas es porque lo que se etiqueta son los elementos llamados **wire** y hemos de poner uno en cada una de las entradas, salidas y nodos que pretendamos etiquetar.

Tras estas acciones el esquema queda de la siguiente manera:

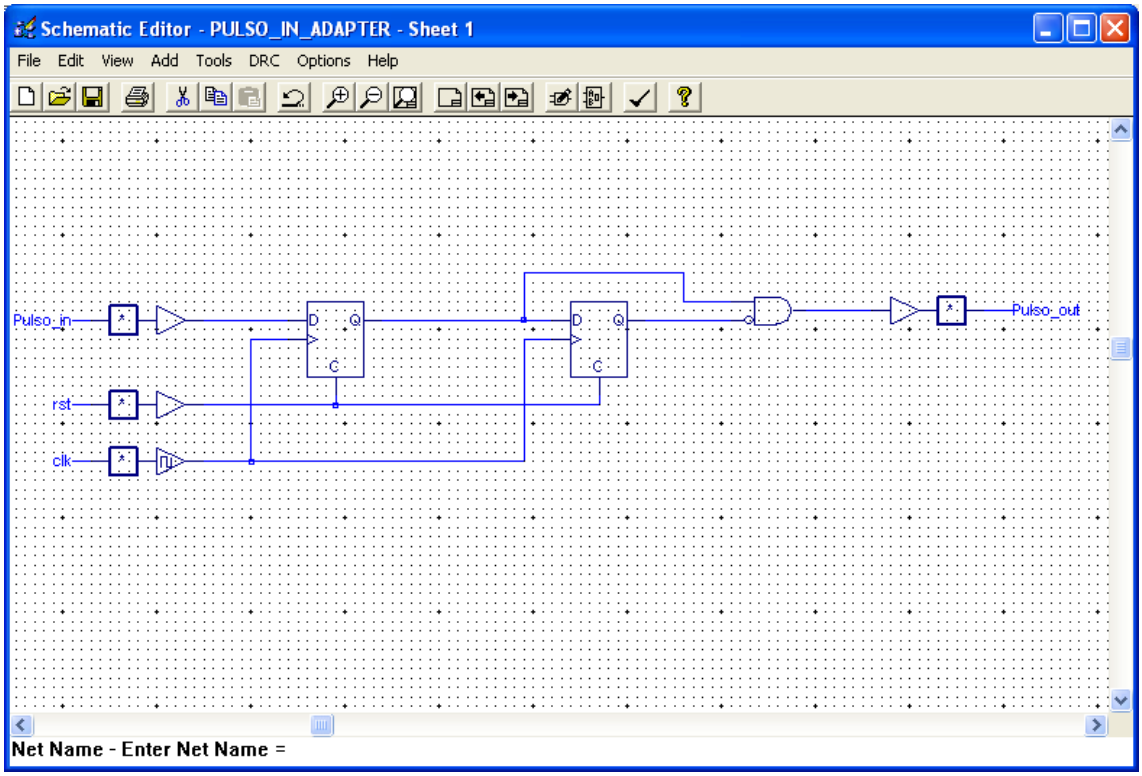


Figura II.15

II.2.7.7 Añadir marcadores de entradas y salidas

Los marcadores de entrada/salida son unos indicadores especiales que identifican a un nodo como una señal de entrada al dispositivo, de salida del mismo o bidireccional. Establecen el sentido de las señales e indican qué nodos son accesibles desde el exterior.

Para introducir los marcadores de entradas y salidas, siga el siguiente procedimiento.

1.- Seleccione **Add > I/O Marker**.

2.- Seleccione **Input**.

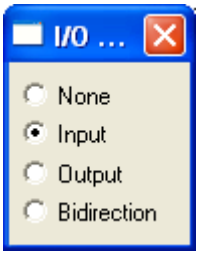


Figura II.16

3.- Pinche y arrastre sucesivas veces para encuadrar cada nodo de entrada con su respectivo nombre y el programa irá añadiendo los marcadores.

4.- A continuación seleccione la opción **Output** de la ventana de diálogo y repita el paso 3 pero esta vez con el nodo de salida.

5.- Cierre la ventana de diálogo.

El esquemático ha de ser similar al de la siguiente figura:

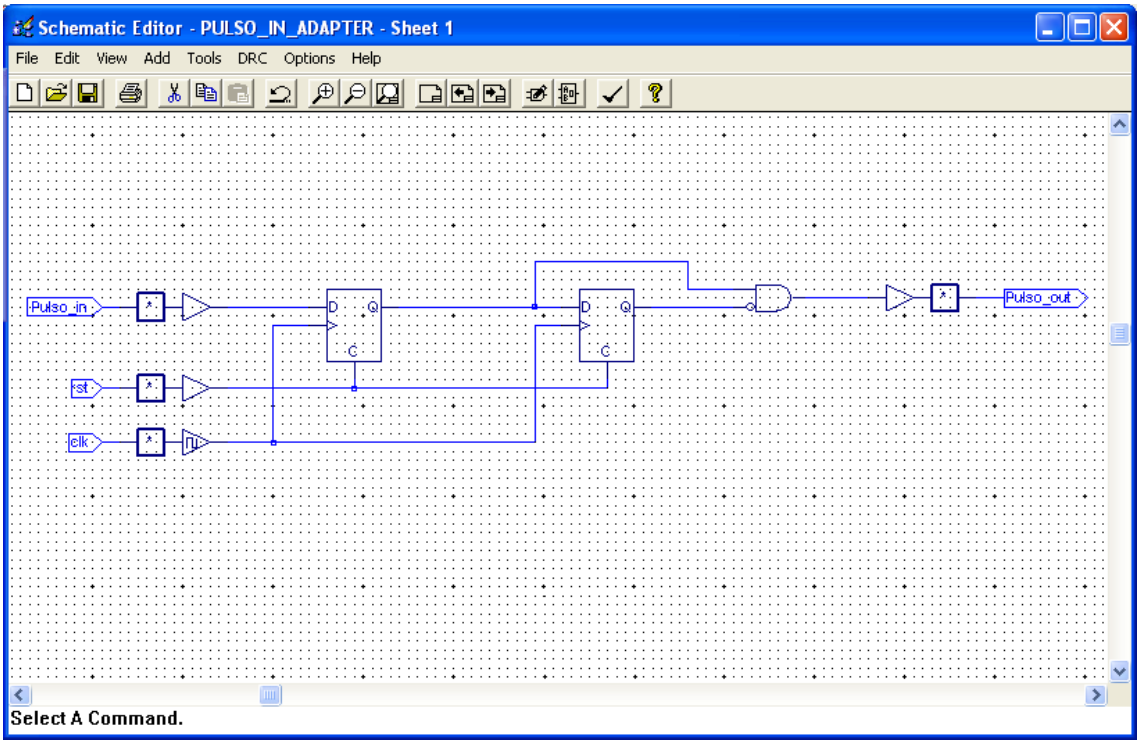


Figura II.17

II.2.8 Creación de un símbolo

El editor de esquemáticos permite crear archivos de símbolos (.sym) mediante el comando **Matching Symbol**. El símbolo que se crea corresponde con el archivo que esté cargado en ese momento y tendrá el mismo nombre pero con la extensión .sym. Los pines de entrada y salida del símbolo tendrán los mismos nombres y sentidos que los marcadores de entrada y salida del esquemático.

El editor de esquemáticos creará el nuevo símbolo en el mismo directorio donde se encuentre el esquemático. Se puede usar el comando **Add Symbol** para insertar el símbolo en cualquier otro esquemático.

Para crear el símbolo del esquema actual, siga el siguiente procedimiento.

1.- Seleccione **File** > **Matching Symbol**, esto crea el símbolo automáticamente.

Para comprobarlo, siga el siguiente procedimiento.

1.- Seleccione **Add > Symbol** para abrir la ventana de diálogo **Symbol Libraries**.

2.- En el cuadro **Library** utilice el scroll para llegar a la primera entrada, llamada “**(Local)**”, y selecciónela. Observe que en el cuadro **Symbol** aparece el símbolo creado anteriormente.

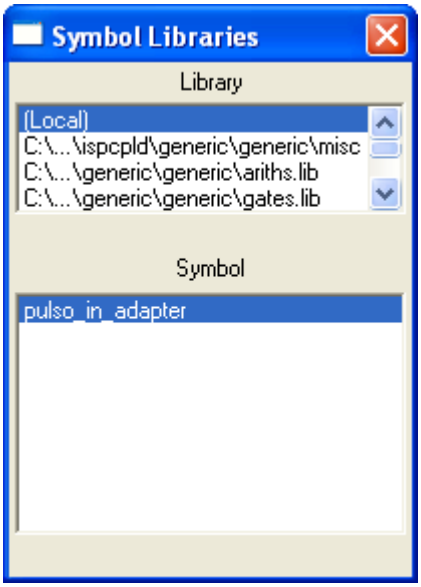


Figura II.18

3.- Cierre la ventana de diálogo.

Para salvar el esquema, siga el siguiente procedimiento.

1.- Seleccione **File** > **Save As**.

2.- Seleccione **Guardar**.

3.- Reemplace el archivo ya existente con el mismo nombre.

II.2.9 Comprobación de las reglas de diseño

El editor de esquemáticos está continuamente chequeando posibles errores, como bucles cerrados y cortocircuitos, mientras se captura el esquema. Pero también se puede realizar una comprobación más exhaustiva cuyos resultados se muestran en un listado, el cual permite, pulsando con el ratón en un error concreto, saltar hasta su ubicación.

Para comprobar los posibles errores de consistencia, siga el siguiente procedimiento.

1.- Seleccione **DRC > Consistency check** para abrir el listado de errores, ventana **Error Report**. No debería haber errores en este momento.

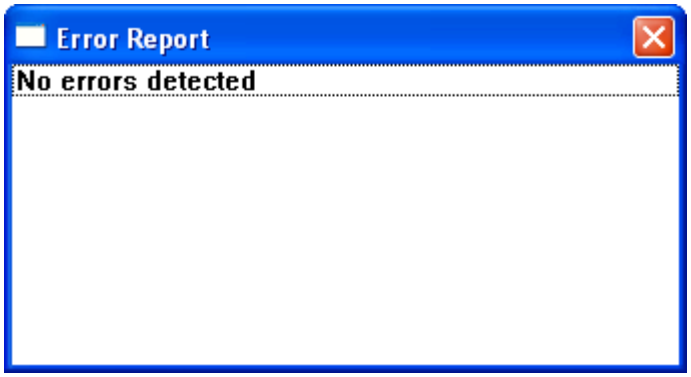


Figura II.19

2.- Cierre la ventana **Error Report**.

3.- Borre algún tramo de cable.

4.- Seleccione **DRC > Consistency check** y observe que esta vez sí hay por lo menos un error en el listado.

5.- Seleccione un error del listado y observe como el editor de esquemas le muestra su situación.

6.- Pulse **ctrl+z** para deshacer el borrado.

7.- Cierre la ventana **Error Report**.

8.- Seleccione **File** > **Exit** para salir del editor de esquemáticos.

Nota: Si pregunta si desea salvar los cambios, indique que no.

II.2.10 Editor de formas de onda

En el entorno ispLEVER Classic 1.2 existen varias formas de definir las formas de onda que más adelante serán usadas para realizar la simulación de nuestros diseños: es posible importar archivos que contengan los vectores de test, definir los estímulos de forma textual o como en el procedimiento que vamos a describir en este documento se pueden definir los estímulos de forma gráfica a través del editor de formas de onda que incorpora la herramienta.

Para editar las señales que intervienen en nuestro diseño, siga el siguiente procedimiento.

1.- En el navegador de proyectos seleccione **Source > New** para abrir la ventana de diálogo **New Source**.

2.- Seleccione **Waveform Stimulus** y pulse el botón **OK**.

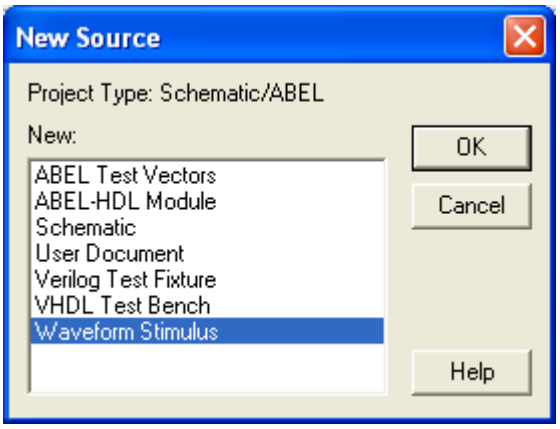


Figura II.20

3.- Se abrirá automáticamente la ventana de diálogo **Associate Waveform Stimulus**, en ella seleccione **pulso_in_adapter** y pulse el botón **OK**, con esto conseguimos que el entorno asocie un nuevo archivo de formas de onda al diseño pulso_in_adapter.

4.- Tras los pasos anteriores se abrirá el editor de formas de onda y aparecerá una ventana donde nos pide un nombre para el archivo, teclee **Pulso_IN_Adapter** y pulse el botón **OK**.

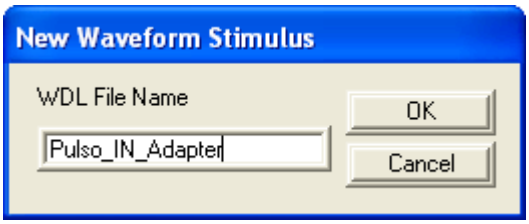


Figura II.21

5.- Seleccione **Options > Timing Options…**; en la ventana **Timing Options** seleccione **1.0** y **ns**, tal y como aparece en la figura II.23, sálvelo como configuración por defecto pulsando el botón **Save As Default** y cierre la ventana **Timing Options**.

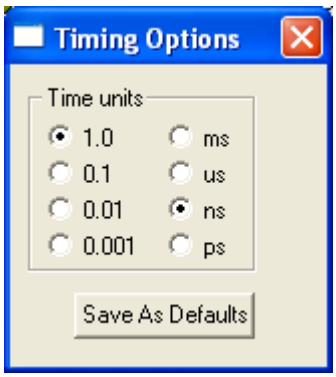


Figura II.21

6.- Seleccione **Edit > Import Wave…**; seleccione las entradas del diseño, pulse el botón **Add** y, a continuación, pulse el botón **Show** y cierre la ventana **Import**.

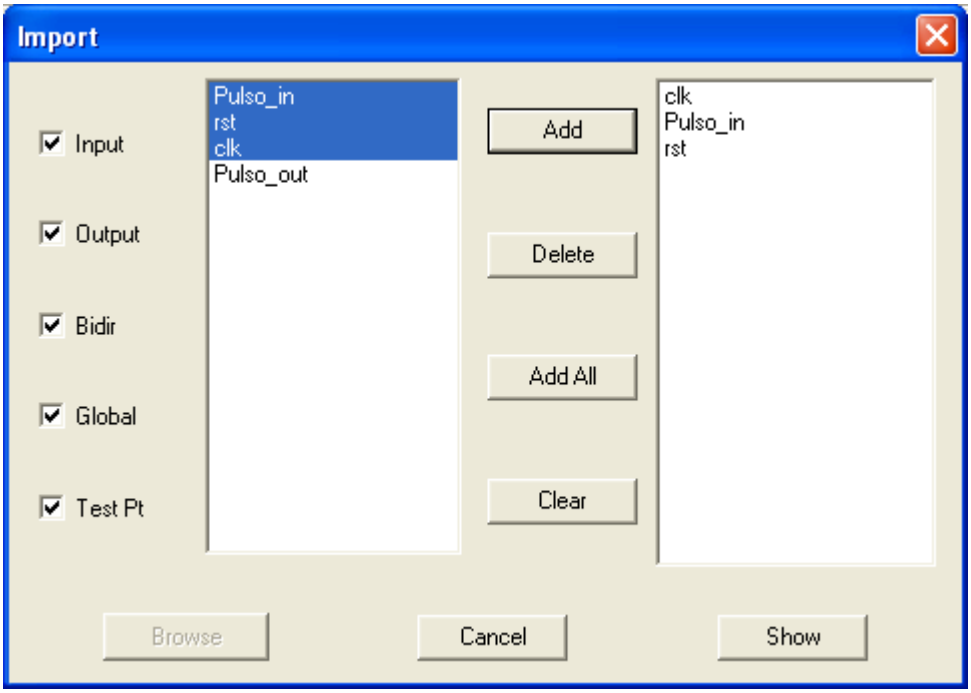


Figura II.22

Ya tenemos listas para ser editadas las señales de entrada a nuestro diseño (clk, Pulso_in, rst)

7.- Pulse **Object > Edit Mode**, y aparecerá la ventana de diálogo de la figura: no la cierre.

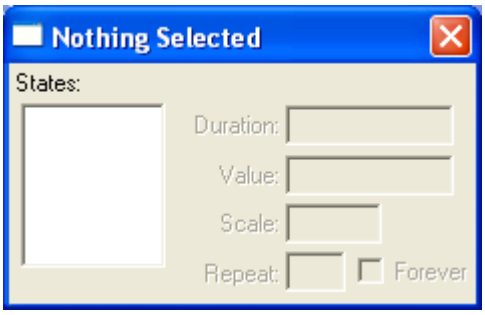


Figura II.23

8.- Pulse sobre el nombre de la señal de reloj del sistema, **clk**, de esta manera queda seleccionada, apreciándose un ligero rayado detrás de su nombre. Pulse a la derecha del nombre de la señal, aproximadamente a un centímetro de distancia, y verá como la ventana se completa con unos valores semejantes a los de la figura II.26.

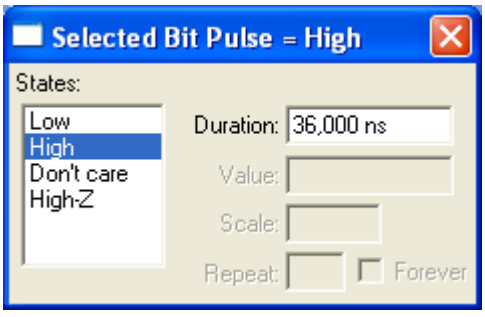


Figura II.24

9.- En el campo **Duration** indique una duración de **20.000 nanosegundos** (valor 20,000ns). Pulse **intro**.

10.- Pulse a la derecha de la forma de onda que se ha dibujado tras el paso anterior e indique de nuevo una duración de 20.000 nanosegundos. Con esto tenemos definido el primer ciclo de la señal de reloj.

11.- Pinche y arrastre sobre la forma de onda dibujada, de modo que se seleccione toda ella, y en la ventana de diálogo aparezca lo siguiente:

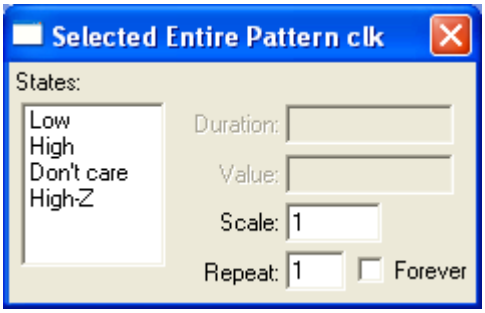


Figura II.25

12.- Seleccione la casilla **Forever**, para generar un reloj de duración indefinida.

13.- Seleccione la señal **Pulso_in** y vaya pinchando repetidamente a la derecha de su nombre y cada vez más lejos, note como en el punto de cada pulsación la señal dibujada alterna entre los valores alto y bajo, **Low** y **High**.

14.- Seleccione la señal **rst**, pinche a su derecha para dibujar una primera porción de onda, a continuación pinche sobre dicha porción. En la ventana que aparece seleccione **Options > Edit Mode**, seleccione **Don’t care** para indicar que el valor inicial de la señal rst no nos importa.

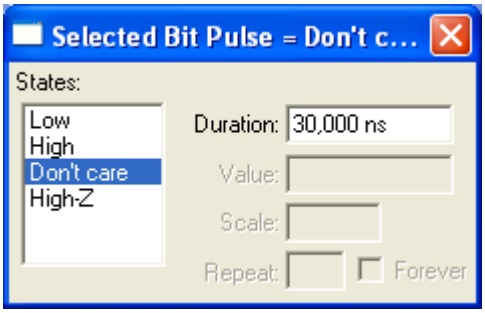


Figura II.26

15.- Pinche un par de veces a la derecha de la onda dibujada para la señal **rst** de tal modo que se defina un pulso parecido al de la figura II.29.

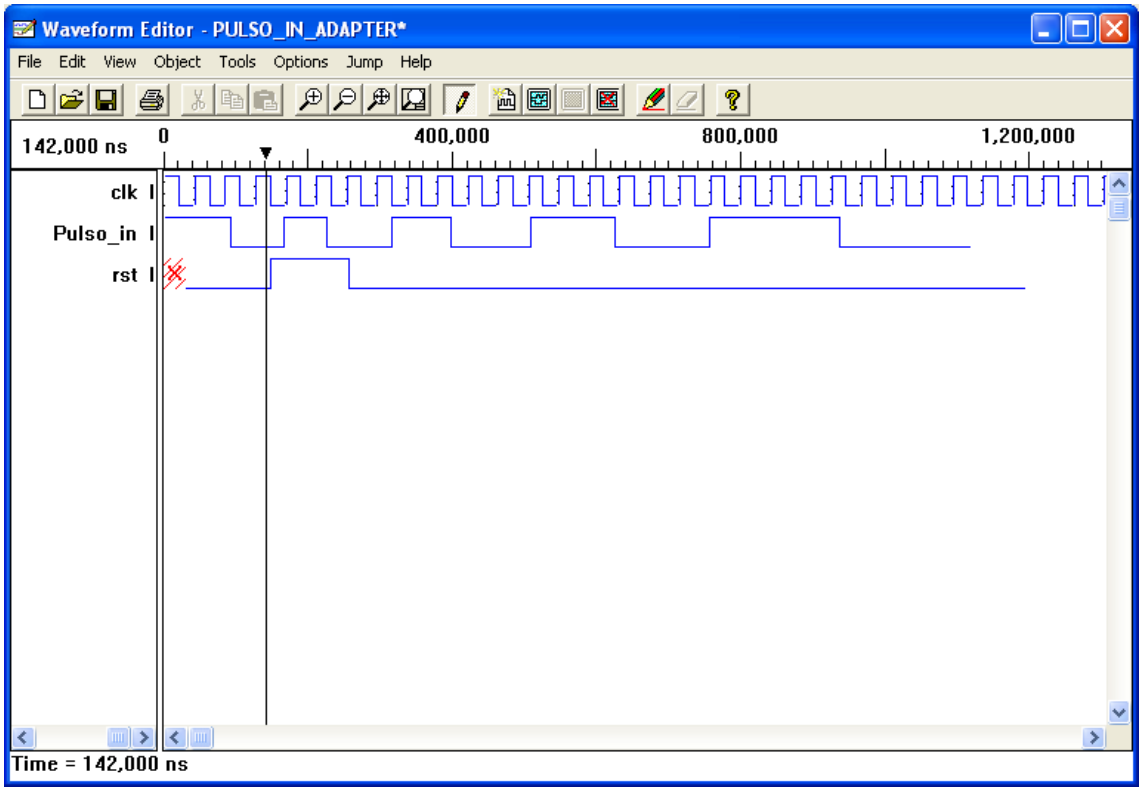


Figura II.27

16.- Guarde el archivo ahora que ya están definidas las formas de onda de las señales de entrada.

17.- Por último cierre el editor de formas de onda.

II.2.11 Realización de una simulación funcional

La simulación funcional es un proceso que permite la detección de errores en la realización del diseño lógico, sirve para realizar una verificación previa a la compilación del diseño. En una simulación funcional no se tienen en cuenta los tiempos de propagación.

Una simulación funcional satisfactoria indica que “en principio” el diseño se comportará de la forma esperada.

Para realizar la simulación funcional de nuestro diseño, siga el siguiente procedimiento.

1.- En el navegador de proyectos seleccione el archivo editado con el editor de formas de onda, **pulso_in_adapter.wdl**.

2.- En el cuadro central, **Processes for current source**, realice doble clic sobre el texto **Functional Simulation** (esto ejecuta una simulación funcional que resulta errónea), se abrirá la siguiente ventana:

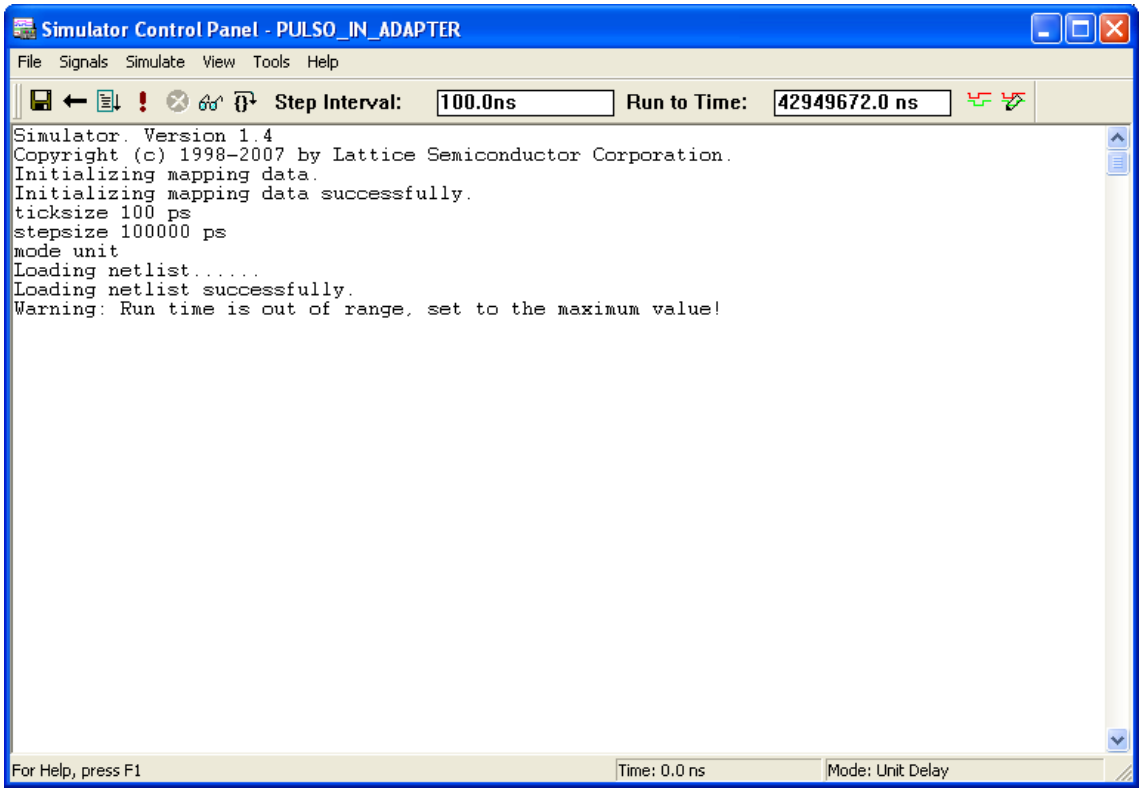


Figura II.29

3.- Para evitar el **Warning** que aparece, introduzca el valor **1000000** en el campo **Run to Time** y pulse sobre la exclamación roja (botón de ejecución). En esta ocasión la simulación funcional se realiza y la ventana anterior queda de la siguiente manera:

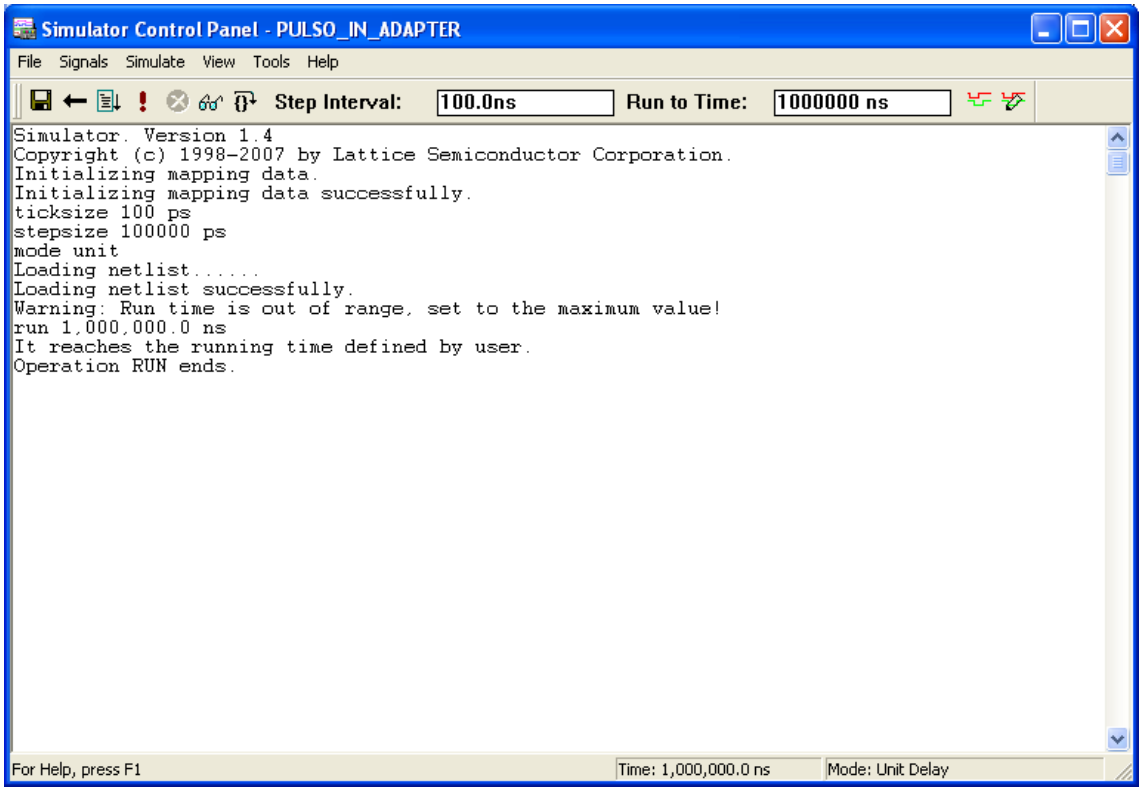


Figura II.28

Tras esto se abrirá el visor de formas de onda, en el cual aparecerán, además de las entradas de nuestro diseño, la salida con su forma de onda.

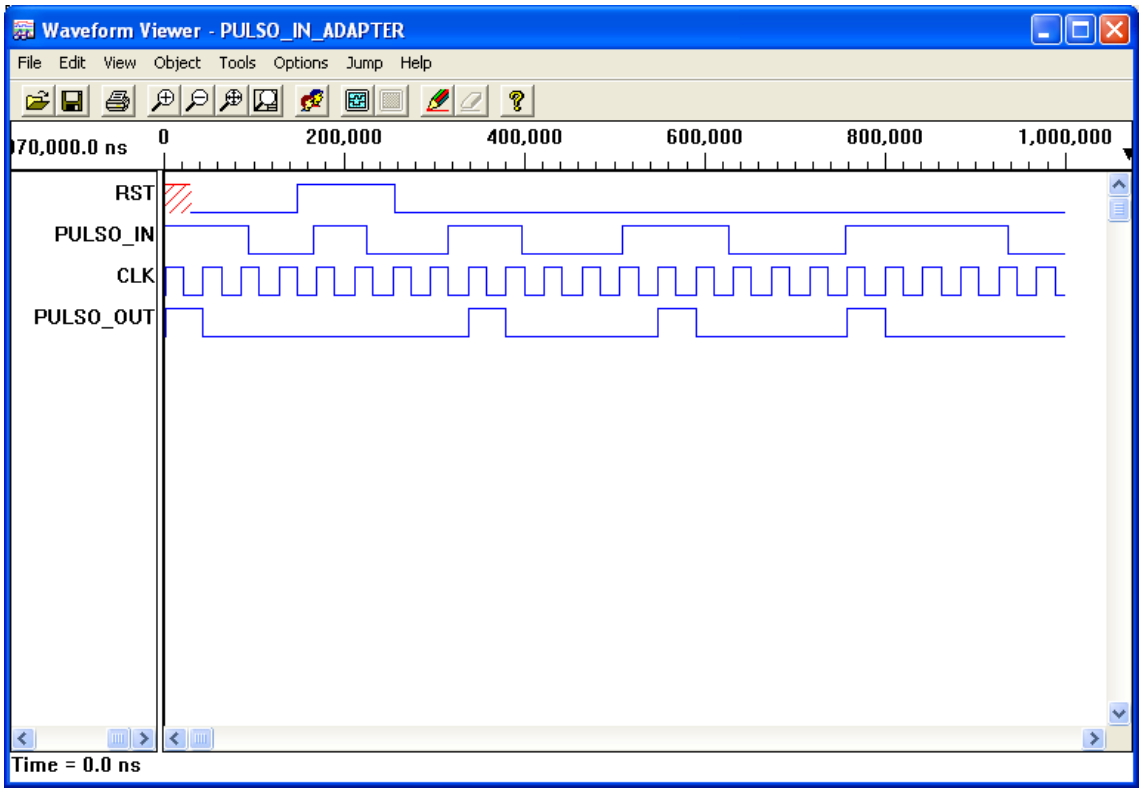


Figura II.29

4.- Observe que el diseño se comportará “en principio” de la forma esperada. Guarde este archivo antes de cerrar las ventanas del visor de formas de onda y del panel de control de simulación. Guarde también el proyecto antes de dar por terminada la parte de captura del esquema y simulación lógica.

II.3 IMPLEMENTACIÓN DEL DISEÑO

II.3.1 Introducción

En el entorno ispLEVER Classic 1.2 la implementación del diseño es un proceso que se realiza en varios pasos que normalmente incluyen la definición de las restricciones deseadas por el usuario, compilación, optimización y adecuación del diseño. Estos pasos individuales se realizan de forma automática y secuencial cuando se ejecuta el proceso **Fit Design** en el navegador de proyectos.

II.3.2 Asignación de pines

En multitud de ocasiones se desea especificar ciertas asignaciones antes de ejecutar el proceso que adecúa el diseño al dispositivo elegido. Estas limitaciones pueden ser, por ejemplo, la localización de ciertos pines, reservar pines, definir el slew rate de las salidas, los niveles de alimentación… todos estos valores se pueden definir mediante el editor de restricciones, **Constraint Editor**, a través de su interfaz gráfico. Si las restricciones definidas no son aplicables en el dispositivo elegido, el editor de restricciones las mostrará en rojo.

Para asignar restricciones al diseño, siga el siguiente procedimiento.

1.- En el navegador de proyectos seleccione el dispositivo elegido en la ventana **Sources in Project**.

2.- En la ventana **Processes for Current Source**, haga doble clic en el texto **Constraint Editor** para que éste se abra.

3.- Despliegue todo el árbol del cuadro de la izquierda, quedando como en la figura.

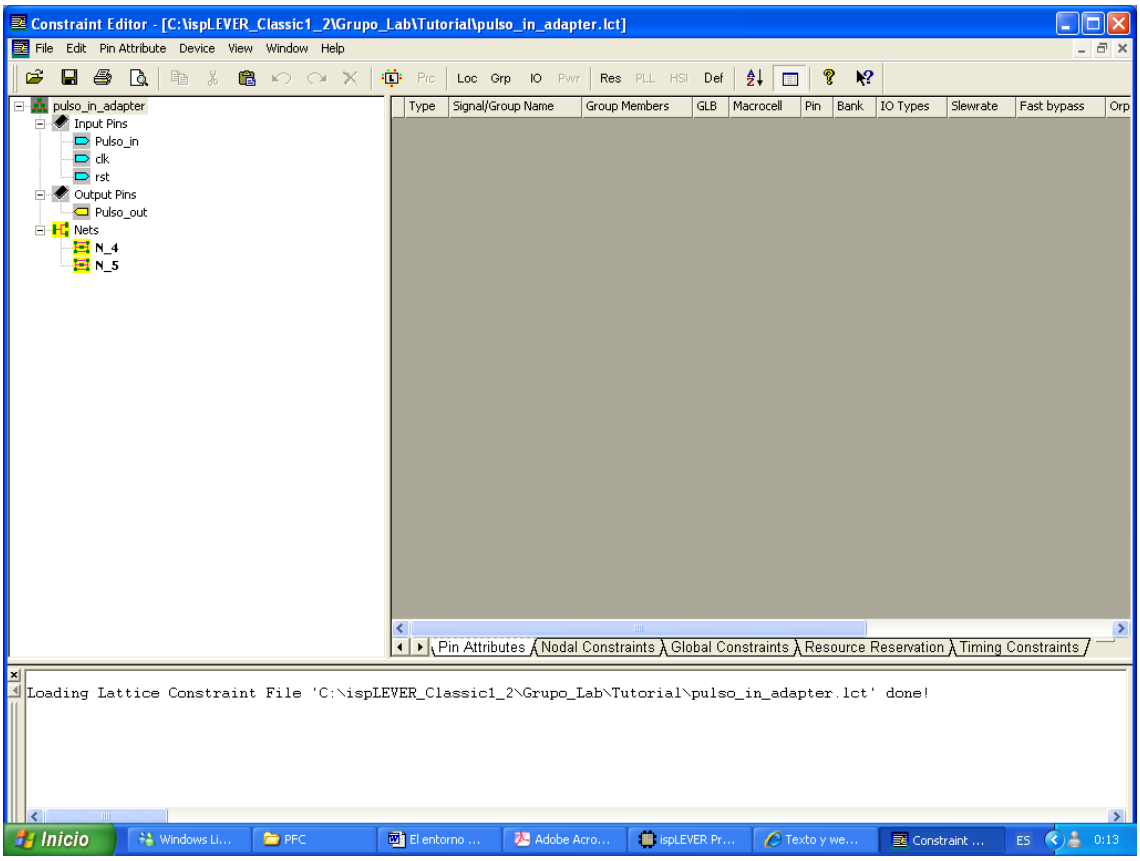


Figura II.30

4.- Seleccione **Pin Attribute > Location Assignment**, para abrir la ventana de diálogo, y siga el siguiente procedimiento.

a.- Bajo **Signals List**, seleccione **Pulso_in**.

b.- Bajo **Assigment Pin**, seleccione el **4**.

c.- Pulse el botón **Add**.

La ventana de diálogo quedará como en la figura.

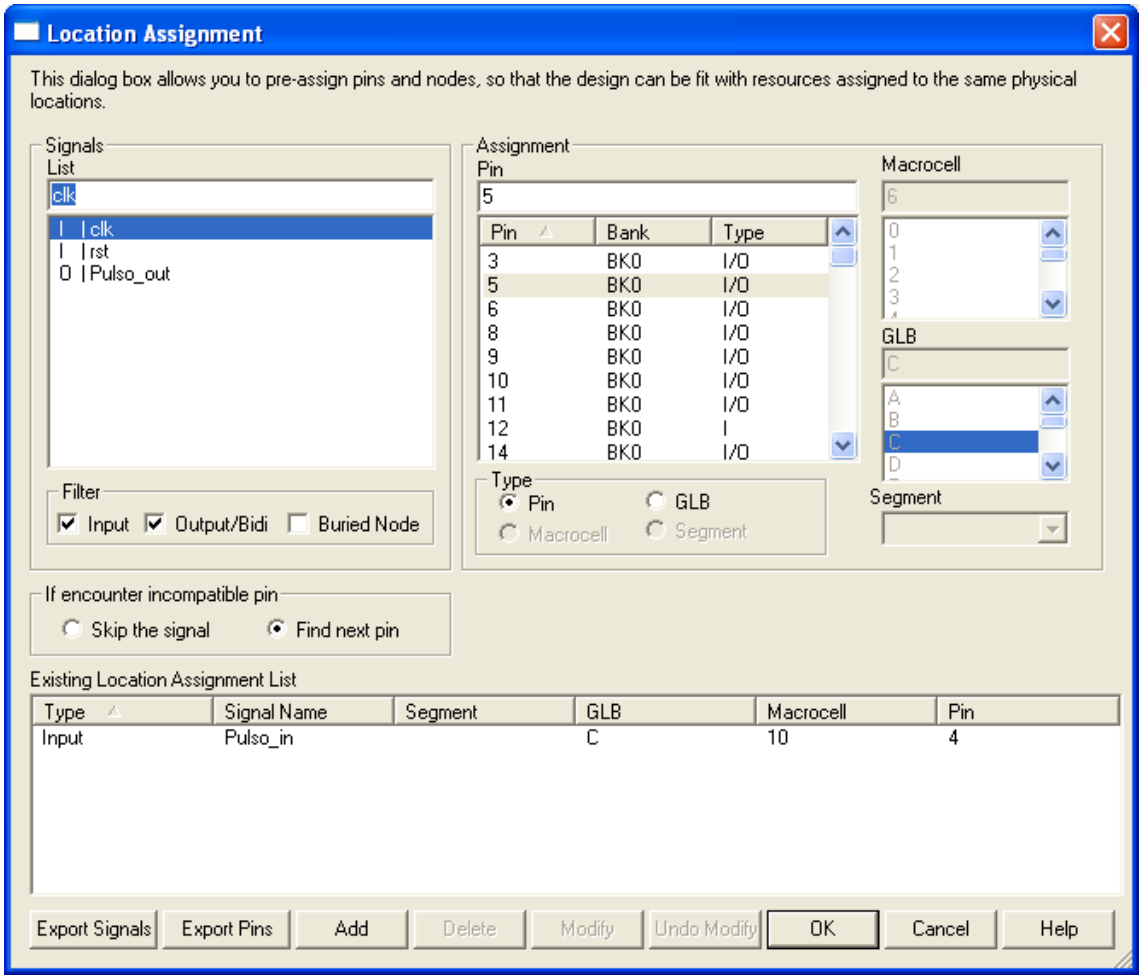


Figura II.31

5.- Pulse **OK** para cerrar la ventana de diálogo.

En el Constraint Editor aparecerá la información del pin asignado en la pestaña Pin Attributes. Y, en la lista de señales, aparecerá una marca junto al nombre de la señal Pulso_in que indica que dicha señal tiene un pin asignado.

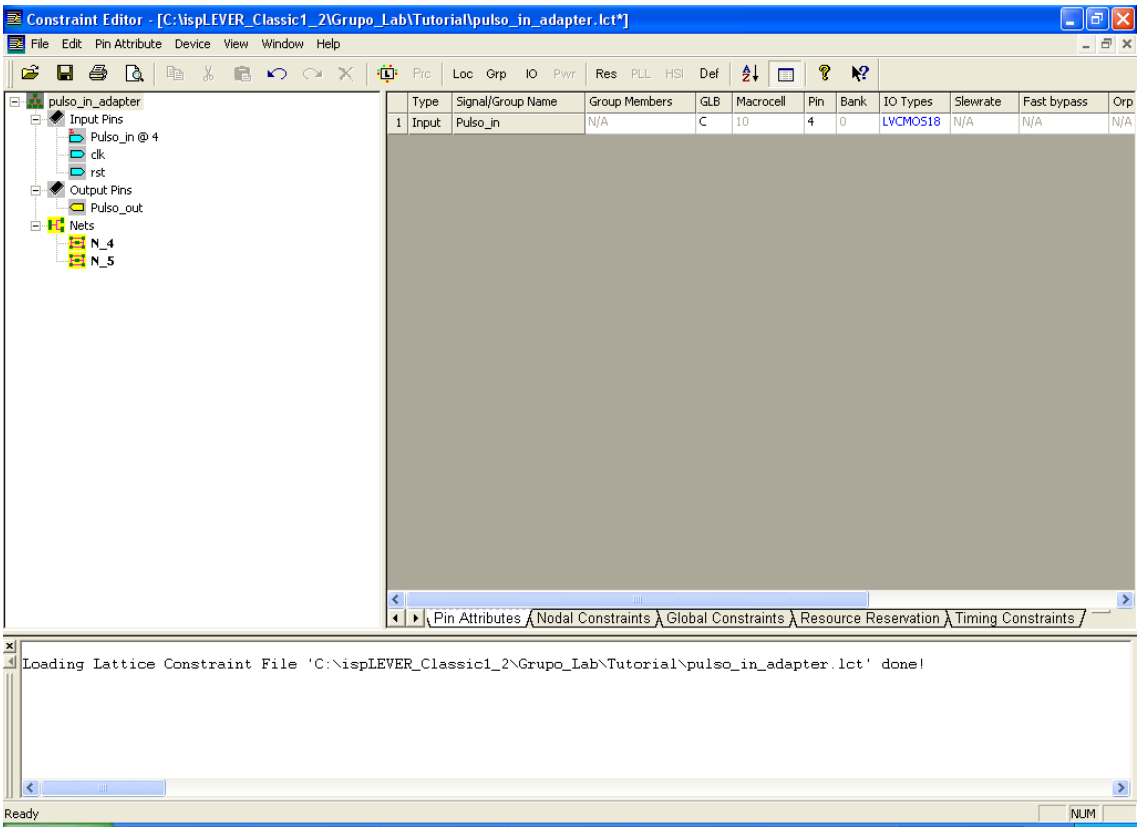


Figura II.32

Otra forma de asignar pines (o de ver los que están asignados) es usar la ventana Package View del Constraint Editor. En dicha ventana los pines se asignan pinchando y arrastrando; existe el siguiente código de colores para definir los pines:

Gris: pin del sistema.

Lima: pin reservado.

Azul: pines de entradas asignados.

Amarillo: pines de salida.

Magenta: pines bidireccionales.

Blanco: pines sin usar.

Para asignar pines gráficamente, siga el siguiente procedimiento.

1.- En el **Constraint Editor** seleccione **Device > Package View** para abrir la ventana **Package View**.

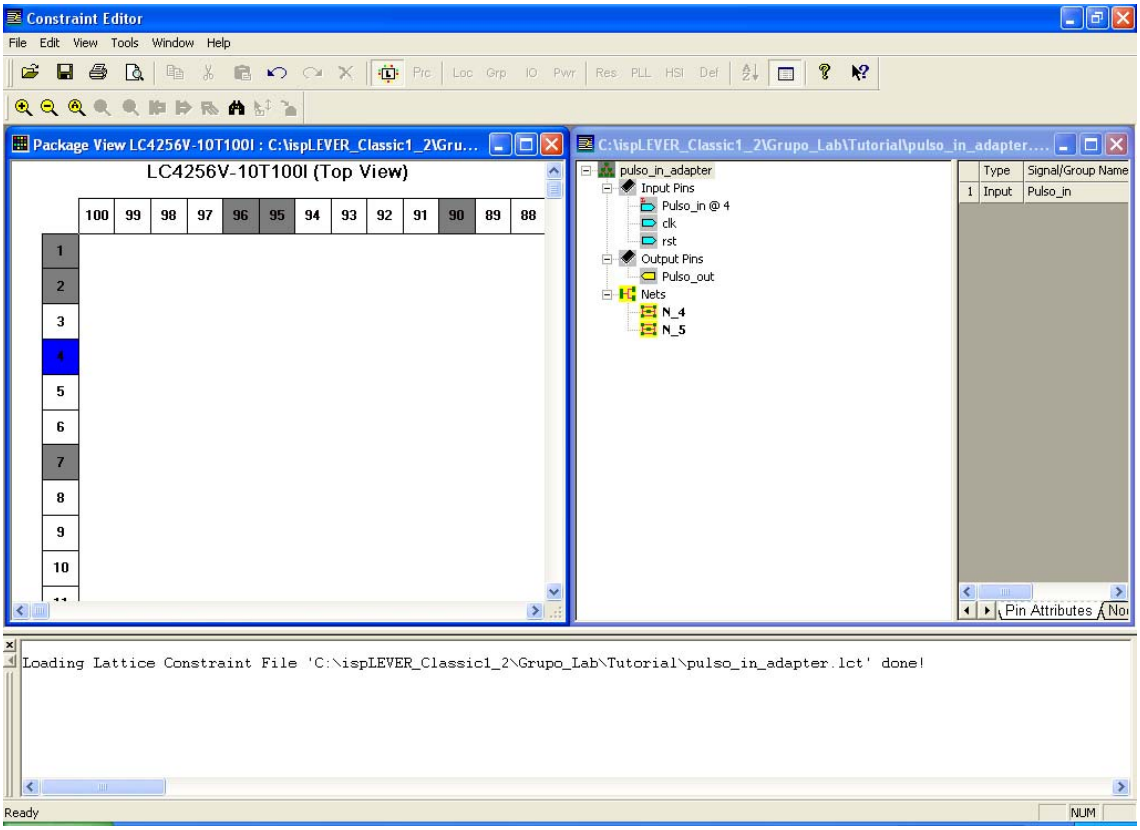


Figura II.33

2.- Para localizar el pin asignado con mayor rapidez, pulse el botón de búsqueda (cuyo icono es unos prismáticos,). En la ventana de diálogo, seleccione **Device Pin** y teclee un **4** en el campo **Name**; finalmente pulse el botón **Find**.

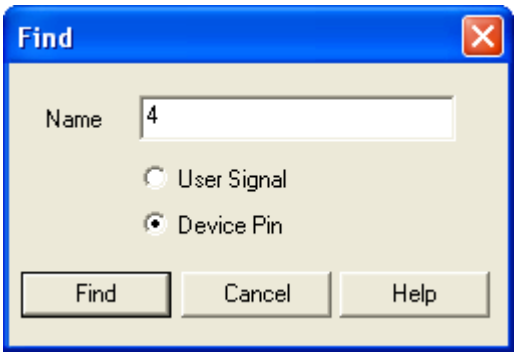


Figura II.34

La vista cambia a la localización especificada.

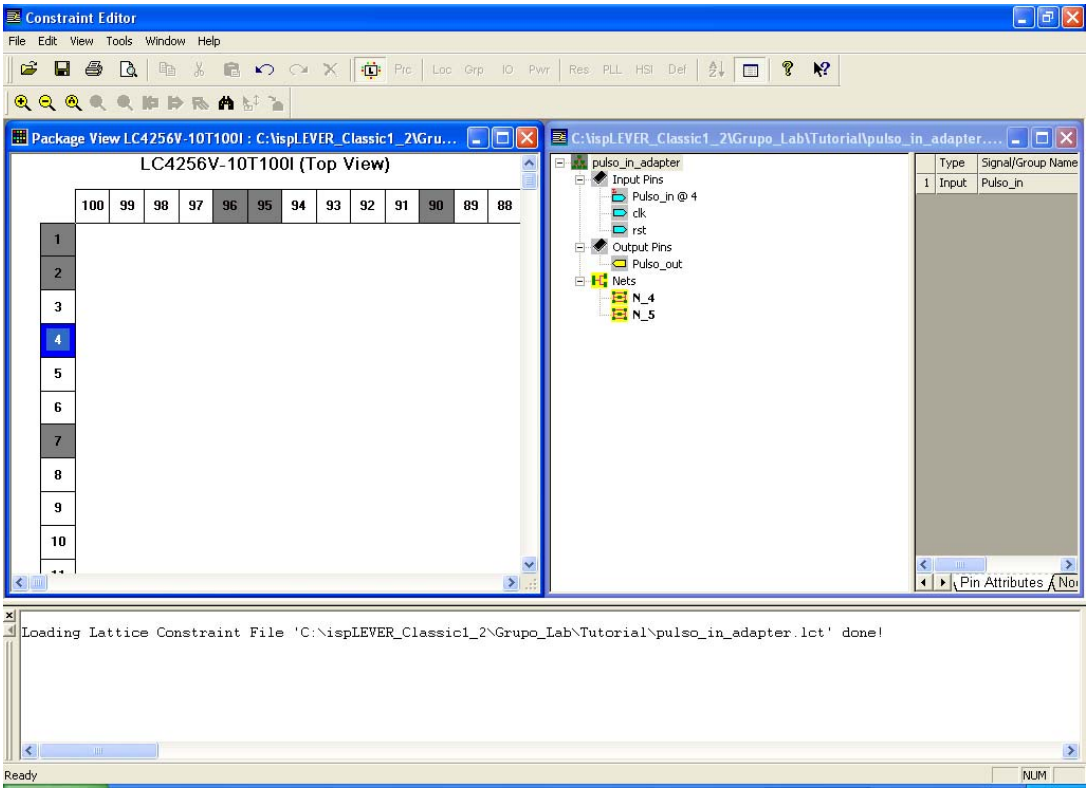


Figura II.35

3.- Si la ventana de la derecha no aparece, seleccione **Window > Tile Verticaly**.

4.- Pinche sobre el nombre de la entrada **rst** y arrastre hasta el pin número 100, con esto queda asignado el pin 100 a la entrada **rst**.

5.- Salve en el **Constraint Editor** y cierre la ventana **Package View**.

II.3.3 Materialización del diseño

El software de ispLEVER tiene un interfaz sencillo, a través del cual adapta el diseño al dispositivo elegido, para obtener el máximo rendimiento posible. Tras una ejecución exitosa del adaptador de diseños (Fitter) el software genera un fichero JEDEC que sirve como fichero de programación.

La adaptación del diseño es un proceso que incluye los siguientes pasos:

Compilación: cambia el formato de entrada del diseño a ecuaciones booleanas, las cuales sirven de entrada para las simulaciones y para los programas de implementación.

Optimización: ejecuta una serie de opciones que permiten alcanzar el máximo rendimiento posible utilizando el menor número de recursos.

Partición: Separa el diseño en bloques individuales según el dispositivo elegido.

Adaptación: Genera el fichero de programación del dispositivo. Genera el fichero JEDEC.

Para materializar un diseño, siga el siguiente procedimiento.

1.- En el navegador de proyectos seleccione el dispositivo elegido en la ventana **Sources in Project**.

2.- En la ventana **Processes for current source** haga doble clic en el texto **Fit Design**. Aparecerá un mensaje en el panel de salida indicando que el proceso se ejecutó correctamente. Aparecerá un tic verde a la izquierda del texto **Fit Design**.

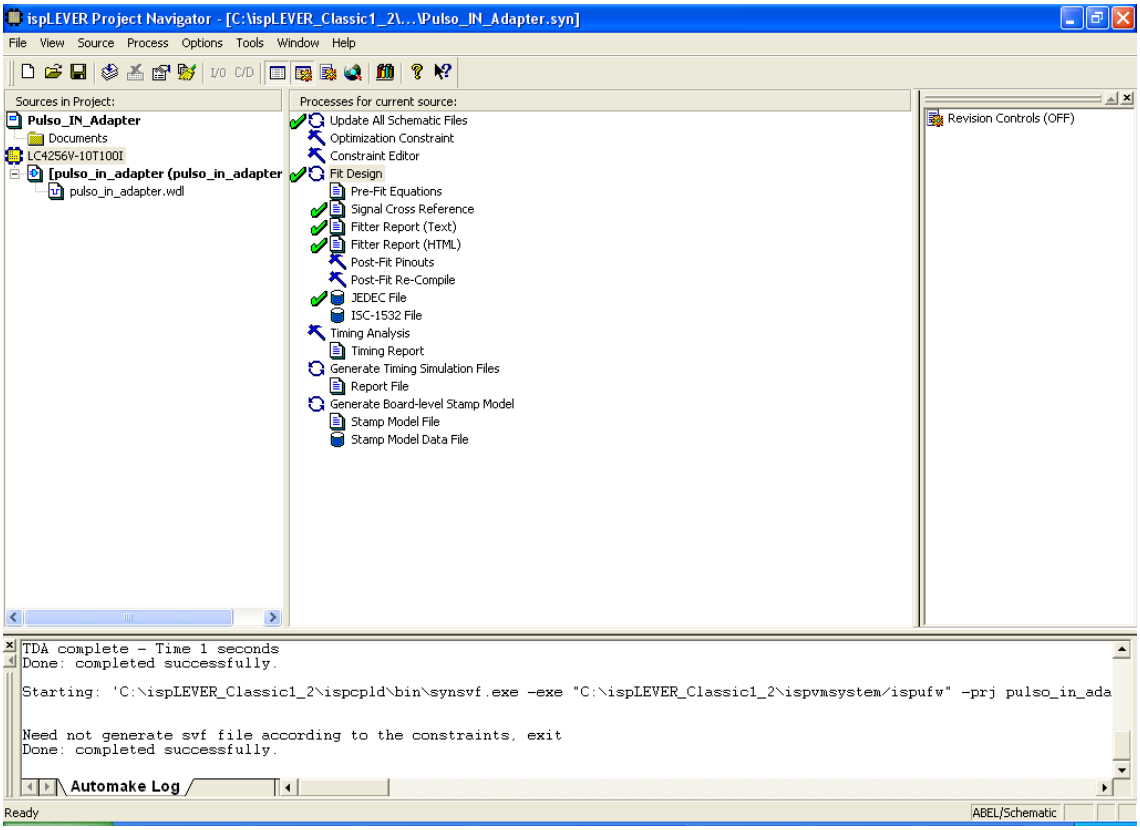


Figura II.36

II.3.4 Configuración de las opciones de visualización de los informes

Por defecto el entorno presenta los ficheros de informes en el panel de salida, pero hay informes que por su interés y extensión es preferible visualizarlos en el visualizador de informes.

Para abrir el visualizador de informes, siga el siguiente procedimiento.

1.- En el navegador de proyectos, seleccione **Options > Environment** para abrir la ventana de diálogo, en ella seleccione la pestaña **Log** y en ésta marque **Using Report Viewer**. Pulse **Aceptar** para cerrar la ventana de diálogo.

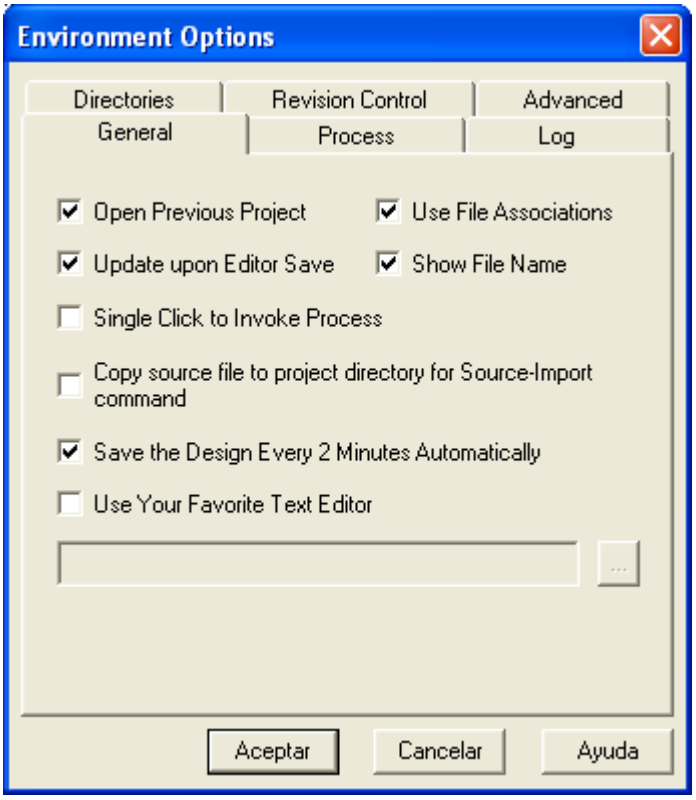


Figura II.39

II.3.5 Lectura del informe de adaptación

El informe de adaptación presenta información estadística como la utilización de pines o el uso de recursos internos del dispositivo.

Para ver el informe de adaptación, siga el siguiente procedimiento.

1.- En la ventana **Processes for current source** haga doble clic en el texto **Fitter Report**, éste se abrirá en el visor de informes. Realice una observación del mismo y cierre su ventana.

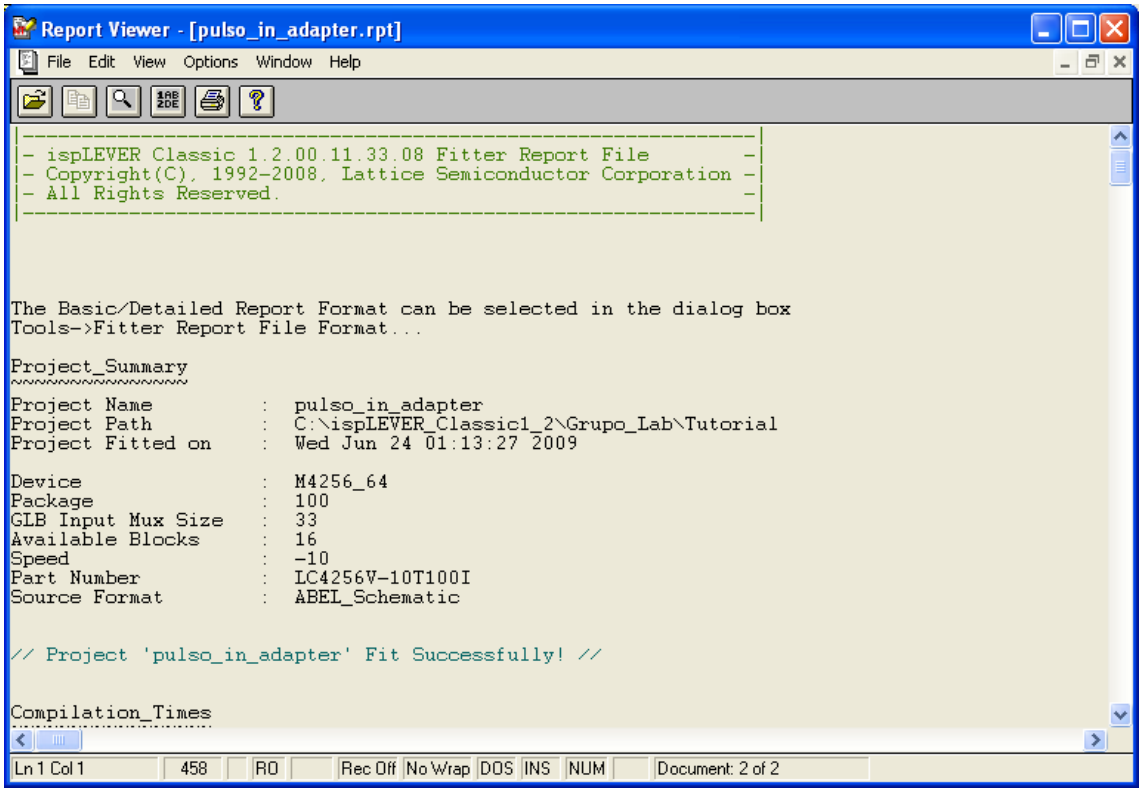


Figura II.37

Como habrá notado, el informe está dividido en múltiples secciones y puede llegar a ser bastante extenso, lo cual dificulta la búsqueda de información concreta, como puede ser la asignación de un determinado pin.

Una alternativa es ver el informe de adaptación usando el navegador HTML; este procedimiento tiene la ventaja de poder buscar usando los vínculos a las distintas secciones del informe que presenta en el panel de navegación.

2.- En la ventana **Processes for current source** haga doble clic en el texto **Fitter Report HTML** para que se abra en el navegador.

3.- Haga clic en el enlace **Pinout Listing** del navegador. Encuentre el **pin 4**, busque a la derecha hasta encontrar el nombre de la señal **Pulso_in**, que es lo que indicamos como restricción definida por el usuario.

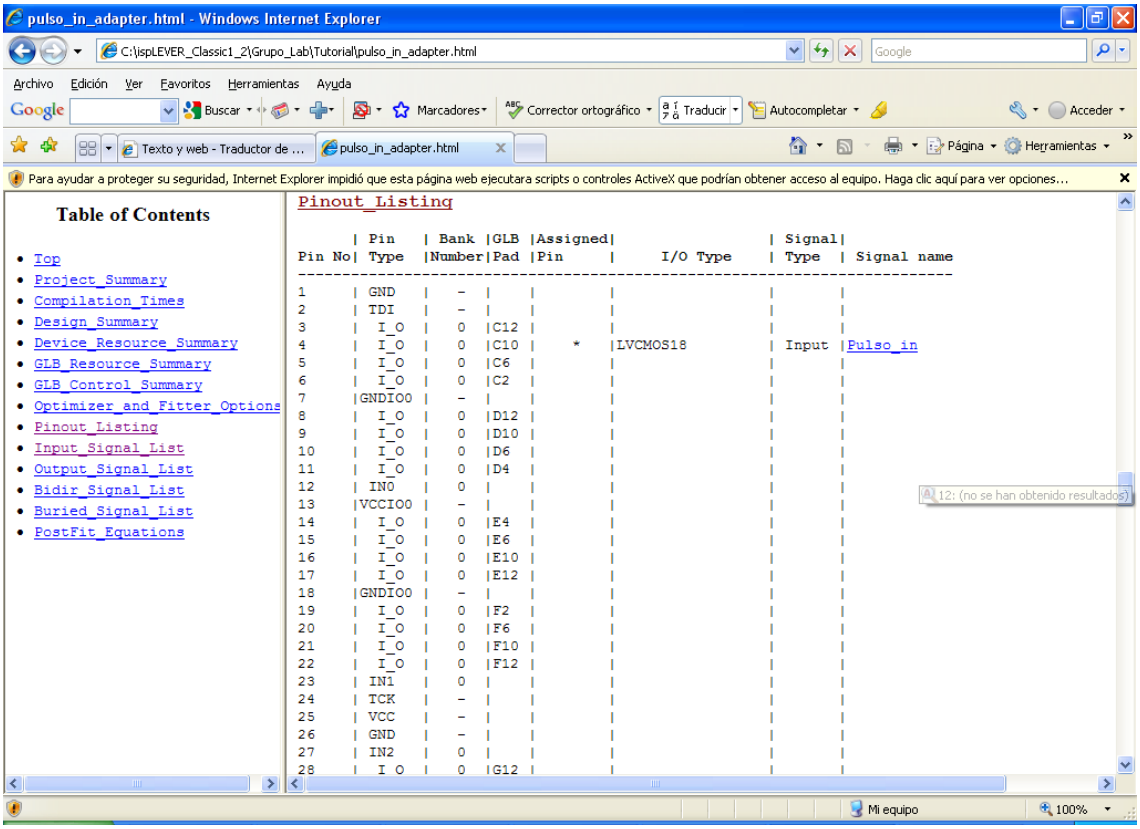


Figura II.38

4.- Cuando termine de examinarlo cierre el navegador.

5.- Seleccione **File** > **Save** para guardar el diseño.

II.4 VERIFICACIÓN DEL DISEÑO

II.4.1 Introducción

El entorno ispLEVER Classic 1.2 ofrece dos posibilidades para realizar la verificación del diseño. La primera opción es el análisis estático de tiempos, mediante el cual obtendremos los parámetros temporales estáticos de nuestro diseño, como son, el camino crítico, los tiempos de set-up y hold y la máxima frecuencia de reloj entre otros. La segunda opción es la simulación con retardos, a cuya salida se presentará la forma de onda de las señales de nuestro diseño. Formas de onda que podrán ser revisadas y medidas utilizando el editor de formas de onda para realizar la revisión de los resultados de la simulación con retardos. Por último, el entorno ispLEVER permite ver en el esquemático, la situación de las señales indicadas en un punto concreto del visor de formas de onda, a lo que se ha llamado correlación de los resultados de la simulación y el esquemático.

II.4.2 Análisis estático de tiempos

El análisis estático de tiempos permite verificar los tiempos de un circuito, con todos los retardos de propagación a lo largo de todos los caminos entre elementos gobernados por el reloj (síncronos) o entre elementos combinacionales. Este análisis permitirá obtener los siguientes resultados: el camino crítico, los requerimientos de tiempos de set-up y hold y la máxima frecuencia de reloj a la que podrá funcionar nuestro diseño.

Una parte del análisis estático de tiempos se encarga de calcular el retardo para cada camino lógico, utilizando para ello un modelo de tiempos del

dispositivo, con el peor caso especificado por el fabricante en las hojas de características del dispositivo.

Los resultados del análisis de tiempos se muestran en una tabla donde en el eje vertical se presentan los orígenes de las señales y en el eje horizontal se presentan los destinos de las mismas. Si entre un origen y destino hay más de un camino, en la celda correspondiente se mostrará el peor caso, es decir, el mayor retardo entre todos los caminos presentes. Para una fácil detección de cuellos de botella se puede hacer doble clic en una celda para ver en detalle todos los retardos del camino.

Para realizar el análisis de tiempos, siga el siguiente procedimiento.

1.- En la ventana **Sources in Proyect** del navegador de proyectos seleccione el dispositivo.

2.- En la ventana **Processes for current source** haga doble clic sobre el texto **Timing Analysis** para arrancar el analizador de tiempos.

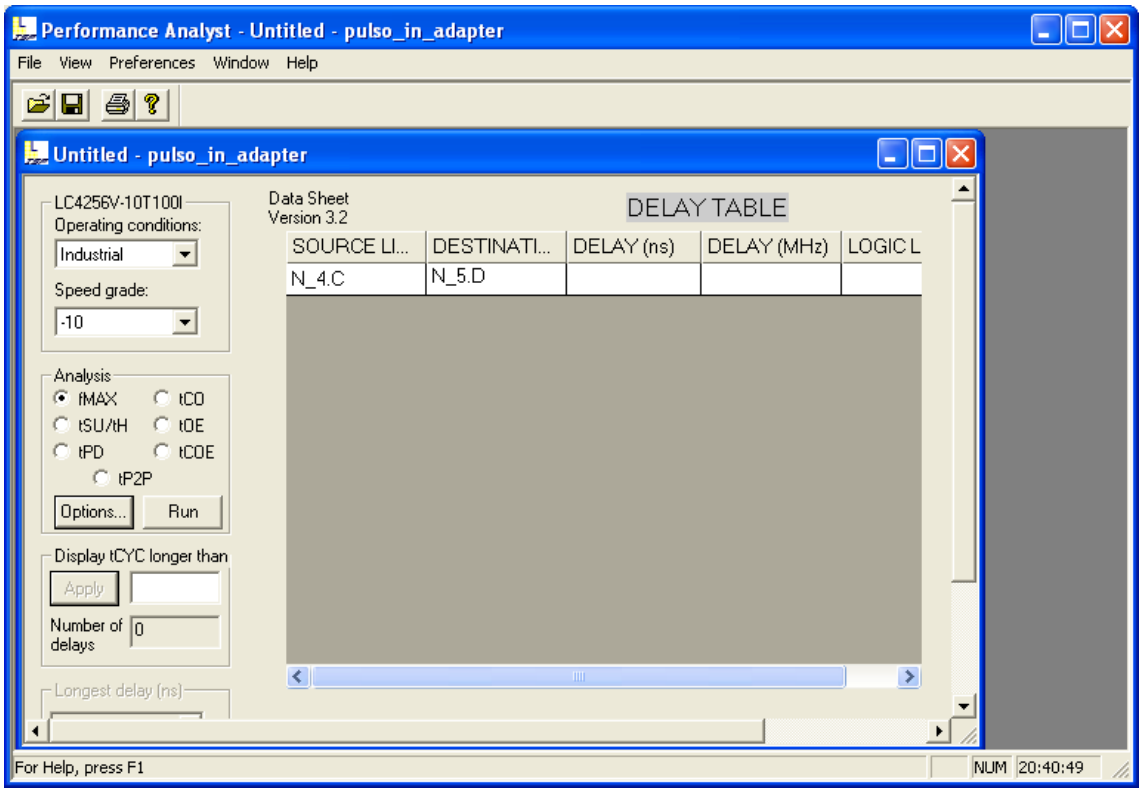


Figura II.39

El analizador presenta siete tipos de análisis distintos: fMAX, tSU/tH, tPD, tCO, tOE, tCOE y tP2P. El primero de ellos, fMAX, es un análisis registro a registro de los retardos internos, mide la máxima frecuencia de reloj, calculándola a partir del peor retardo entre registros. El análisis tP2P analiza el camino entre dos puntos indicados por el usuario. Los otros cinco tipos de análisis se centran en los retardos entre pines externos. Se pueden utilizar filtros con umbrales de tiempo, de origen y destino y de nodo para ajustar más cada análisis.

3.- En el cuadro llamado **Analysis**, seleccione la opción **tCO**.

4.- Pulse el botón **Run** para realizar el análisis. Los resultados del mismo pueden verse en la figura II.44.

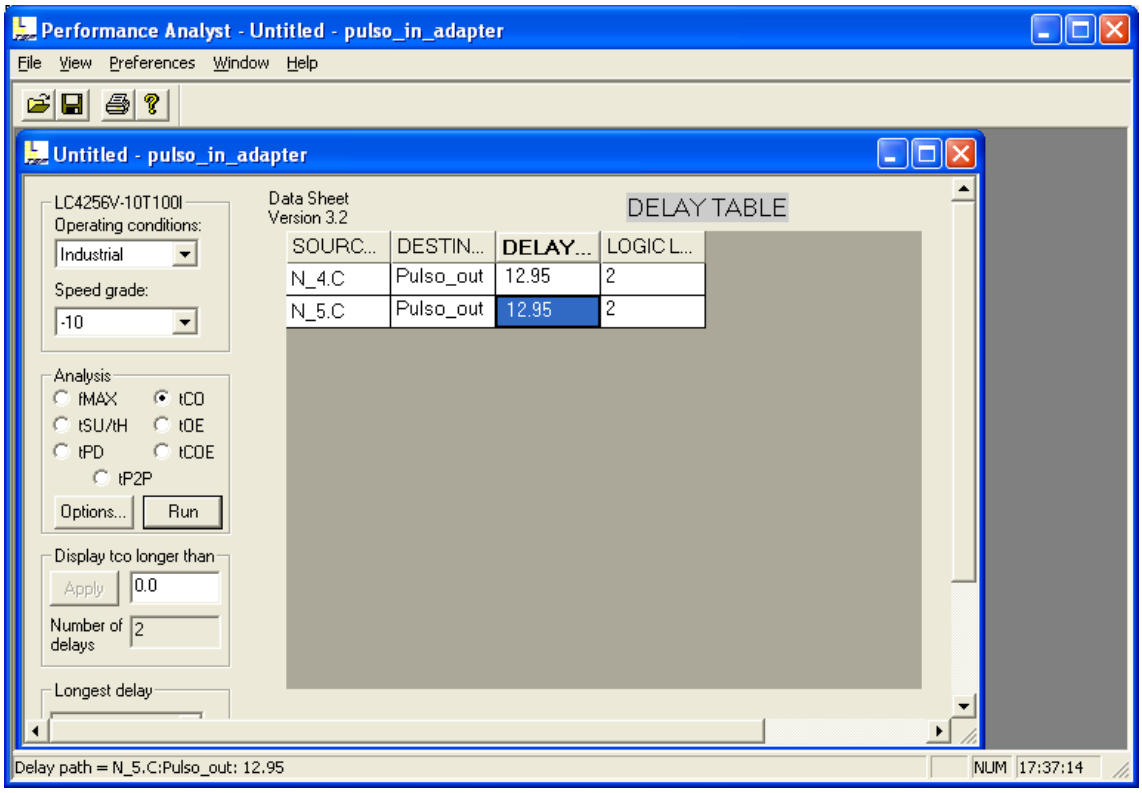


Figura II.40

En este caso, el retardo según este análisis es de 12,95 ns.

5.- Haga doble clic en la celda con fondo azul y se abrirá la ventana **Expanded Path**, donde se presenta la información detallada del path con mayor retardo.

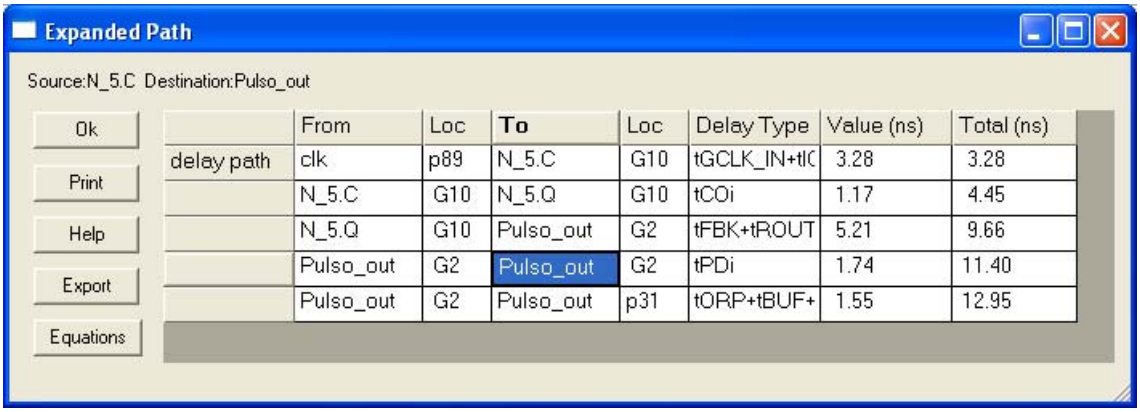


Figura II.41

6.- Haga clic en el botón **Equations** para abrir la ventana con el mismo nombre, en la que se muestra la relación funcional entre la pareja fuente/destino seleccionada.

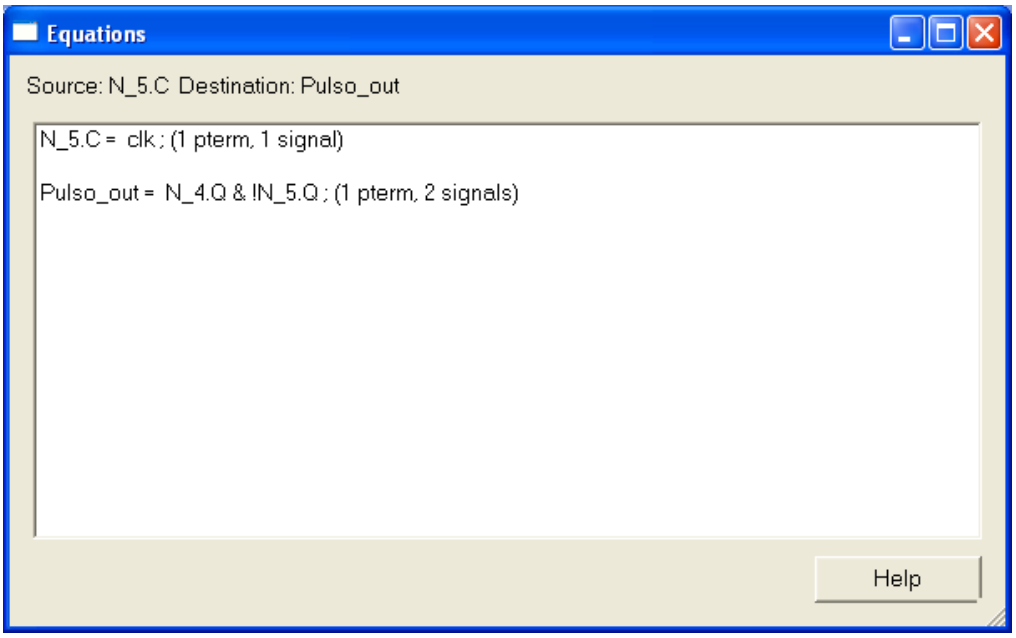


Figura II.42

7.- Cierre el **Performance Analyst** sin guardar.

II.4.3 Simulación con retardos

En pasos anteriores se utilizó el simulador lógico de Lattice para realizar simulaciones funcionales, pero ahora se utilizará para realizar una simulación con retardos.

La simulación con retardos difiere del análisis de tiempos en un par de puntos, por un lado para realizar una simulación con retardos se necesita el archivo con los vectores de test y, por otro lado, aunque ambas herramientas proporcionan información de tiempos de retardo, solo el simulador lógico simula el diseño lógico, ofreciendo como salida un gráfico con las formas de onda. En consonancia con esto, la herramienta que uses dependerá de las necesidades concretas de cada diseño. Se usará el análisis estático de tiempos para revisar de forma rápida los caminos críticos y el simulador lógico se utilizará para un análisis más detallado a través de la simulación.

Para realizar una simulación con retardos, lo primero que hay que hacer es generar un archivo .abv que contenga los vectores de test.

Para crear el archivo de vectores de test, siga el siguiente procedimiento.

1.- En el navegador de proyectos seleccione **Source > New**, esto abrirá la ventana de diálogo **New Source**.

2.- Seleccione la opción **ABEL Test Vectors**

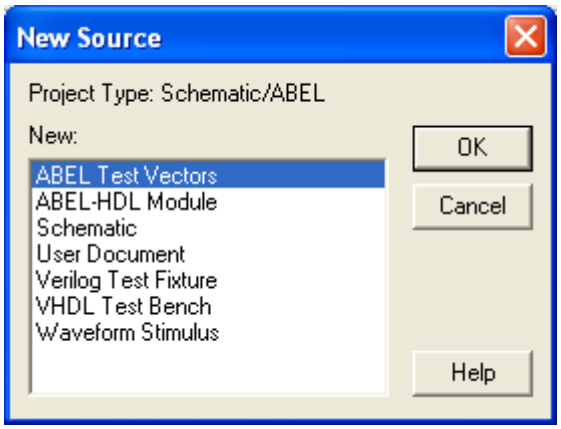


Figura II.43

3.- Pulse el botón **OK**.

4.- Introduzca el nombre del archivo en el cuadro **File Name** de la ventana que se ha abierto (**New File**). En esta ocasión el archivo lo llamaremos **test** con la extensión que indica por defecto (**.abv**).

El editor de texto del entorno se abrirá con un documento en blanco cuyo nombre es el que hemos indicado.

5.- Edite el siguiente texto en el documento de texto:

```
module pulso_in_adapter;
rst pin;
pulso_in pin;
clk pin;
pulso_out pin;
test_vectors
([clk,pulso_in,rst]->[pulso_out])
```

```
[0,0,0]->[0];
[1,0,0]->[0];
[0,1,0]->[0];
[1,1,0]->[1];
[0,0,1]->[0];
[1,0,1]->[0];
[0,1,1]->[0];
[1,1,1]->[0];
END
```

6.- Guarde el archivo. Con esto hemos creado el archivo **.abv**.

7.- Cierre las ventanas del editor de texto.

Para realizar una simulación con retardos, siga el siguiente procedimiento.

1.- En la ventana **Sources in Proyect**, del navegador de proyectos, seleccione el archivo **test.abv** que es el que contiene la información de los vectores de test.

2.- En la ventana **Processes for current source** haga doble clic sobre el texto **Timing Simulation**. Esto abrirá el panel de control del simulador lógico de Lattice. En la barra de herramientas asegúrese de que el campo **Step Interval** contiene el valor **10.0ns** y el campo **Run to Time** contiene el valor **1000 ns**.

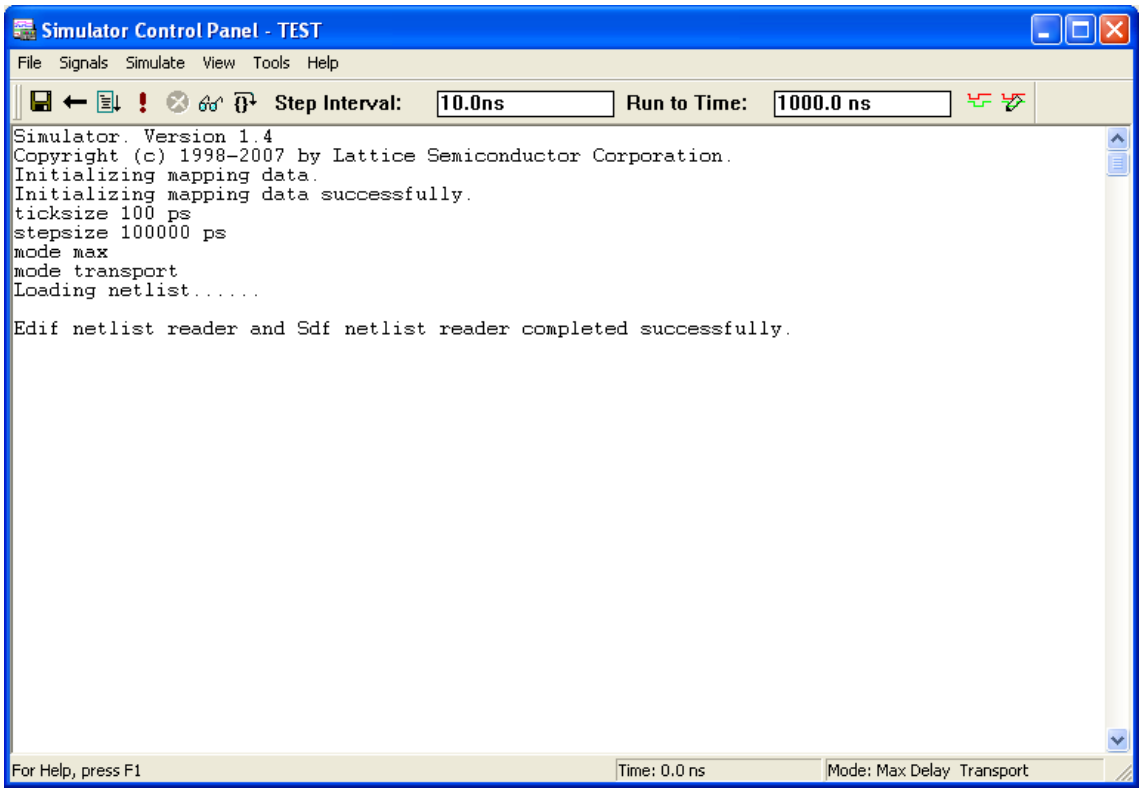

Simulator Control Panel - TEST
File Signals Simulate View Tools Help
Step Interval: 10.0ns Run to Time: 1000.0 ns
Simulator. Version 1.4
Copyright (c) 1998-2007 by Lattice Semiconductor Corporation.
Initializing mapping data.
Initializing mapping data successfully.
ticksize 100 ps
stepsize 100000 ps
mode max
mode transport
Loading netlist.....
Edif netlist reader and Sdf netlist reader completed successfully.
For Help, press F1
Time: 0.0 ns
Mode: Max Delay Transport

Figura II.44

3.- Seleccione **Simulate > Run** para que dé comienzo la simulación con retardos y para que se abra el visor de formas de onda.

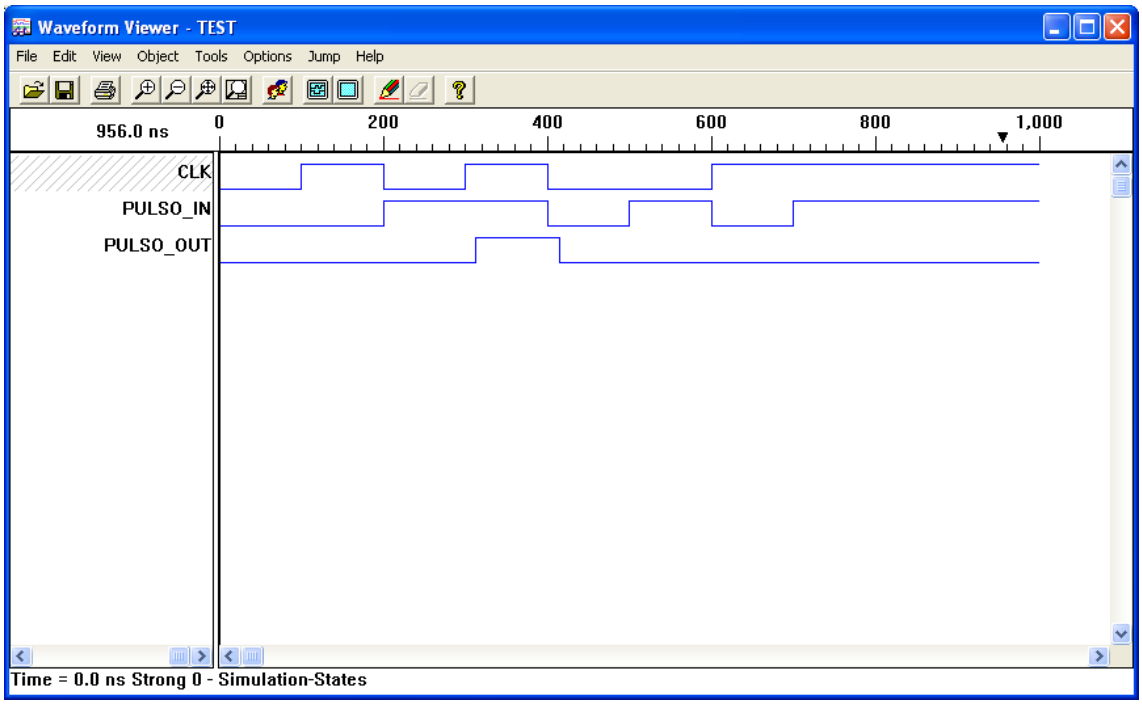


Figura II.45

Para que aparezca la señal RST, siga el siguiente procedimiento.

1.- En el visor de formas de onda seleccione **Edit > Show…** para que se abra la ventana **Show Waveforms**.

2.- Seleccione la señal **RST** y pulse sobre el botón **Show**.

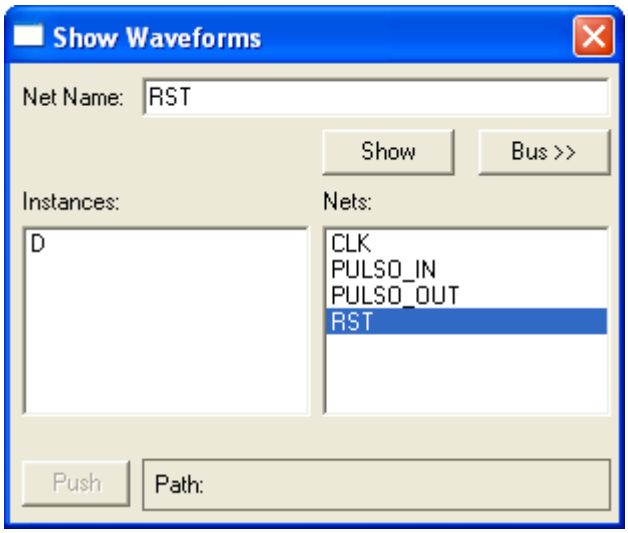


Figura II.49

3.- Cierre la ventana **Show Waveforms**.

4.- Observe que la respuesta del diseño es la esperada y que en esta simulación están presentes los retardos.

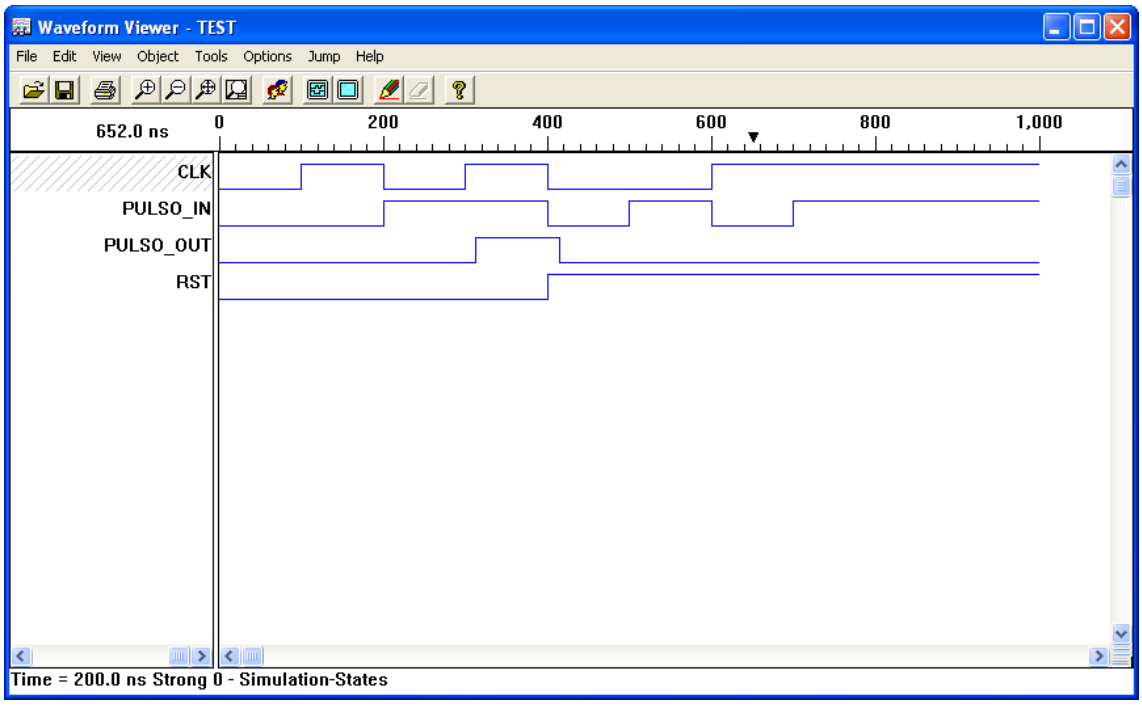


Figura II.46

5.- Cierre la ventana del visor de formas de onda y guarde el resultado cuando el entorno se lo indique.

II.4.4 Revisión de los resultados de la simulación con retardos

Se puede utilizar el visor de formas de onda para medir la diferencia de tiempo entre dos eventos.

Para medir la diferencia entre dos eventos, siga el siguiente procedimiento.

1.- En la ventana **Waveform Viewer**, dentro de la barra de herramientas, seleccione el **Zoom in**. Advierta que el cursor del ratón pasa a ser una Z.

2.- Haga clics sucesivos en las formas de onda para ir acercando la visión. Haga clic con el botón derecho del ratón para salir del modo **Zoom**, el cursor pasa a ser normal.

3.- Seleccione **Jump > Time=0**. Esto le llevará al comienzo de las formas de onda (t=0).

4.- Seleccione la señal **CLK** haciendo clic en su nombre.

5.- Haciendo clic en cualquier punto de las formas de onda aparecerá una línea vertical en todo el visor y en la línea de comandos aparecerá el punto temporal en el que está la línea. Haga clic en un punto.

6.- Seleccione **Jump > Next change**. La marca se desplaza hasta el siguiente flanco de la señal **CLK**.

7.- Seleccione **Object > Place Marker** para dejar una marca permanente en dicho punto.

8.- Ahora seleccione la señal **PULSO_OUT**.

9.- Seleccione **Jump > Next Change**. La marca temporal se desplaza hasta el siguiente flanco en esta señal. El valor **Delta** en la barra de estado nos indica la diferencia temporal entre ambos puntos, en este caso 12,9 ns. Conviene recordar que según el análisis estático de tiempos el retardo era de 12,95 ns, un valor que podemos considerar coincidente con el hallado tras esta simulación con retardos, ya que la diferencia se debe a la falta de resolución.

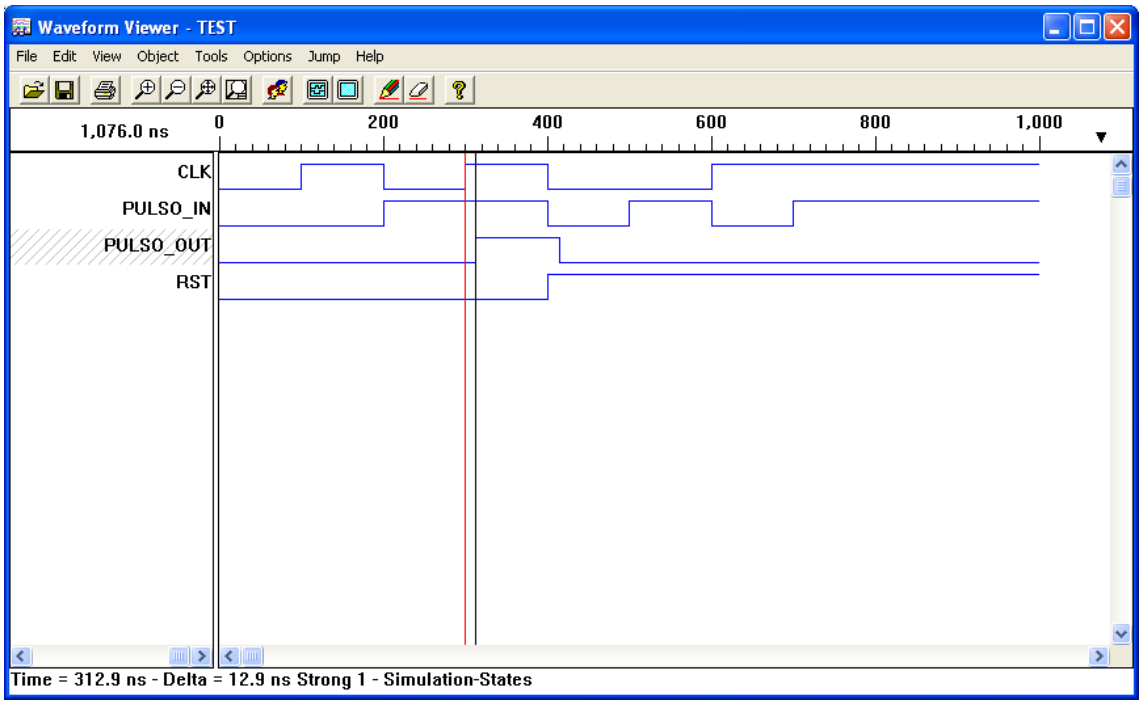


Figura II.47

II.4.5 Correlación de los resultados de la simulación y el esquemático

El entorno permite ver en el esquemático la situación de las señales en un punto concreto del visor de formas de onda.

Para ver en el esquemático la situación de las señales, siga el siguiente procedimiento.

1.- Mientras el visor de formas de onda está en ejecución, seleccione el esquemático en el navegador de proyectos.

2.- En la ventana **Processes for current source**, haga doble clic en **Navigate Hierachy** para abrir el navegador jerárquico. Colóquelo detrás del visor de formas de onda.

3.- En el navegador jerárquico utilice el **Zoom** para ver correctamente el diseño.

4.- En el visor de formas de onda seleccione **Object > Hide Marker**.

5.- Seleccione **Jump > Time=0**.

6.- Seleccione la señal de salida y observe que en el navegador jerárquico cambia de color. Próximo a las etiquetas de los pines de entrada y salida aparecen unos cuadrados donde se indica el valor de cada pin en el momento indicado por la marca del visor de formas de onda. Esto permite ir viendo la evolución de las señales en el navegador jerárquico.

7.- Seleccione **Jump > Next Change** en el visor de formas de onda para comprobar que cambian los valores en el navegador jerárquico.

8.- Cierre el navegador jerárquico, el visor de formas de onda y el simulador lógico de Lattice.

III Capítulo III: “Tecnologías de FPGAs y PLDs del fabricante Lattice”

III.1 INTRODUCCIÓN

En este capítulo se exponen brevemente las características generales de los dispositivos de cada una de las familias, del fabricante Lattice, estudiadas. Para cada familia se comentan los datos generales, se relatan las posibilidades de configuración ofrecidas por el fabricante y se muestra a través de una imagen el esqueleto del código de producto con las variantes que tenga en cada familia. Se ha realizado una separación inicial entre las familias de FGPAs y las familias de PLDs.

En el apartado reservado a las FPGAs se ha realizado la división del mismo atendiendo de forma aislada a cada una de las distintas familias estudiadas.

Para el apartado que trata sobre el estudio de las familias de PLDs se ha realizado una primera división entre CPLDs y SPLDs, según sean familias de dispositivos lógicos programables complejos o simples, respectivamente. Por último, en cada uno de estos apartados se incluye la información referente a cada una de las familias estudiadas.

Gracias a los apartados dedicados a los datos generales de cada familia se pretende que este capítulo pueda servir como guía básica para facilitar la elección de la familia a utilizar en futuros diseños.

Los apartados dedicados a las posibles formas de configuración de cada familia se han incluido porque, aunque en el laboratorio se suele utilizar un único método de configuración, podría resultar interesante que la placa de prototipado permitiese más métodos de programación. Pudiendo así familiarizar a los alumnos con distintos métodos de configuración de dispositivos.

Los apartados en los que se muestra el esqueleto del código de producto, a través de una imagen, se han incluido para que este capítulo pueda servir para conocer el código del dispositivo que se quiera adquirir. También se pretende que estos apartados sirvan para poder consultar las características de un dispositivo una vez que lo tenemos sobre la mesa.

Como último apartado de este capítulo, se incluyen las conclusiones a las que se ha llegado. Dichas conclusiones se refieren a la conveniencia o no, de cada una de las familias de dispositivos para incluir uno de sus miembros como elemento central, para la elaboración de una placa de prototipado; que sería utilizada en un laboratorio de enseñanza de herramientas de CAD.

En estas conclusiones se tiene en cuenta si los dispositivos de cada familia presentan funcionalidades que no serían utilizadas a lo largo de un laboratorio semestral, que parte de un nivel básico como es el del tutorial realizado. Y también se tiene en cuenta si los dispositivos de cada familia serían o no capaces de materializar los posibles diseños de mediana complejidad que podrían ser exigidos.

III.2 FPGAs

III.2.1 Familia de dispositivos LatticeECP3

Para revisar una información más detallada sobre estos dispositivos, consulte el apartado V.1 DISPOSITIVOS LATTICE ECP3.

III.2.1.1 Datos generales

La familia de FPGAs de Lattice ECP3 utiliza la tecnología de lógica reconfigurable SRAM, lógica basada en LUT (lookup table). Los dispositivos de esta familia tienen desde 17K hasta 149K LUTs, además presentan entre 133 y 586 pines de entrada/salida de usuario, contienen bloques de memoria interna sysMEM, PLLs, DLLs, entre 24 y 320 multiplicadores de 18x18, entradas/salidas paralelas estándar y entradas/salidas serie.

Los dispositivos de la familia ECP3 de Lattice ofrecen la posibilidad de establecer comunicaciones serie de alta velocidad, gracias a los bloques SERDES (SERializer/DESerializer) se alcanzan velocidades desde 150 Mbps hasta 3.2Gbps. Utilizando estos bloques se pueden establecer comunicaciones serie basadas en los protocolos: PCI Express, SMPTE, Ethernet (XAUI, GbE y SGMII) y CPRI.

Los dispositivos de esta familia ofrecen entre 12 y 160 sysDSPs, Procesadores Digitales de Señales, diseñados expresamente para conseguir, a bajo coste, un alto rendimiento en el procesado digital de señales.

Los dispositivos de la familia ECP3 tienen un puerto sysCONFIG que permite la configuración tanto serie como paralelo del dispositivo.

III.2.1.2 Configuración

Todos los dispositivos Lattice ECP3 tienen dos puertos que pueden utilizarse para la configuración del dispositivo. El Test Access Port (TAP) soporta configuración bit-wide y el puerto sysCONFIG soporta la configuración dual-byte, byte y serie. Los dispositivos de esta familia permiten la configuración in-system a través del puerto TAP. El puerto sysCONFIG es un interfaz de 20 pines con siete entradas/salidas utilizadas como pines dedicados y el resto como pines de doble uso.

En el momento del encendido la memoria SRAM de la FPGA está preparada para ser configurada a través del puerto sysCONFIG. Una vez seleccionado un puerto de configuración, éste permanece activo hasta que se realiza el ciclo de configuración completo. Tras el arranque, se puede activar el puerto TAP en cualquier momento, enviando el comando apropiado a través del puerto TAP.

Además, cada dispositivo de la familia tiene un puerto JTAG. Esta familia también ofrece un oscilador en el chip y la capacidad de detectar errores software en el flujo de configuración.

III.2.1.3 Dispositivos

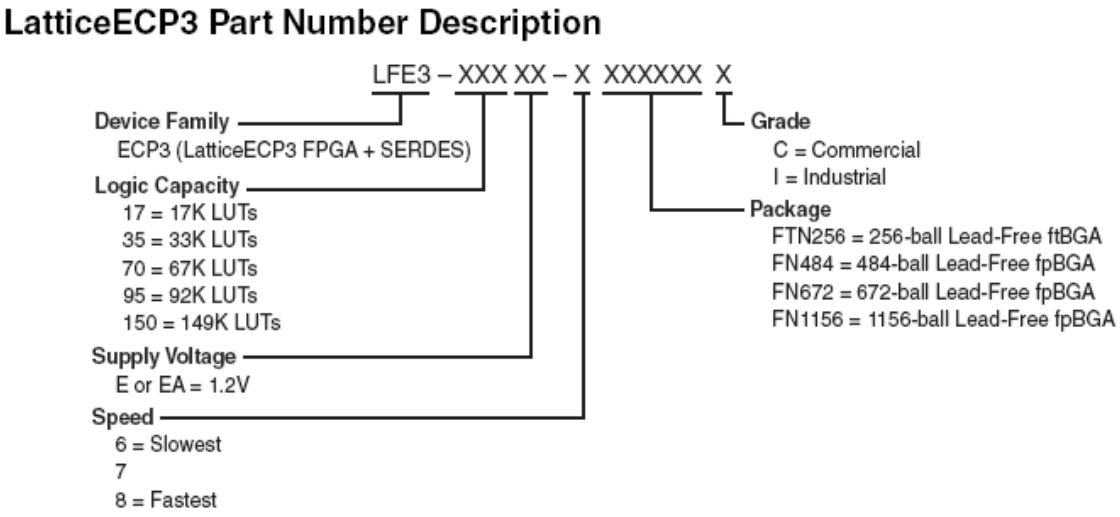


Figura III.1

III.2.2 Familia de dispositivos LatticeECP2/M

Para revisar una información más detallada sobre estos dispositivos, consulte el apartado V.2 DISPOSITIVOS LATTICE ECP2/M.

III.2.2.1 Datos generales

La familia de FPGAs de Lattice ECP2/M utiliza la tecnología de lógica reconfigurable SRAM, lógica basada en LUT (lookup table). Los dispositivos de esta familia tienen desde 6K hasta 95K LUTs, además presentan entre 90 y 583 pines de entrada/salida de usuario, contienen bloques de memoria interna sysMEM, PLLs, DLLs, entre 12 y 88 multiplicadores de 18x18, entradas/salidas paralelas estándar y entradas/salidas serie.

La familia de dispositivos LatticeECP2/M está optimizada para ofrecer características de alto rendimiento como bloques sysDSP, SERDES de alta velocidad (solo en la familia LatticeECP2M) e interfaces de alta velocidad.

Los dispositivos de esta familia ofrecen entre 3 y 42 sysDSPs, Procesadores Digitales de Señales, diseñados expresamente para conseguir, a bajo coste, un alto rendimiento en el procesado digital de señales.

Los dispositivos de la familia ECP2/M tienen un puerto sysCONFIG que permite la configuración tanto serie como paralelo del dispositivo.

III.2.2.2 Configuración

Todos los dispositivos LatticeECP2/M tienen dos puertos que pueden utilizarse para la configuración del dispositivo. El Test Access Port (TAP) soporta configuración bit-wide y el puerto sysCONFIG soporta configuración byte-wide y serie. El puerto sysCONFIG es un interfaz de 20 pines con siete entradas/salidas utilizadas como pines dedicados y el resto como pines de doble uso.

En el momento del encendido la memoria SRAM de la FPGA está preparada para ser configurada a través del puerto sysCONFIG. Una vez seleccionado un puerto de configuración, éste permanece activo hasta la finalización del ciclo de configuración. El puerto TAP se puede activar en cualquier momento tras el arranque, enviando el comando apropiado a través del puerto TAP.

Además, cada dispositivo de la familia tiene un puerto JTAG.

III.2.2.3 Dispositivos

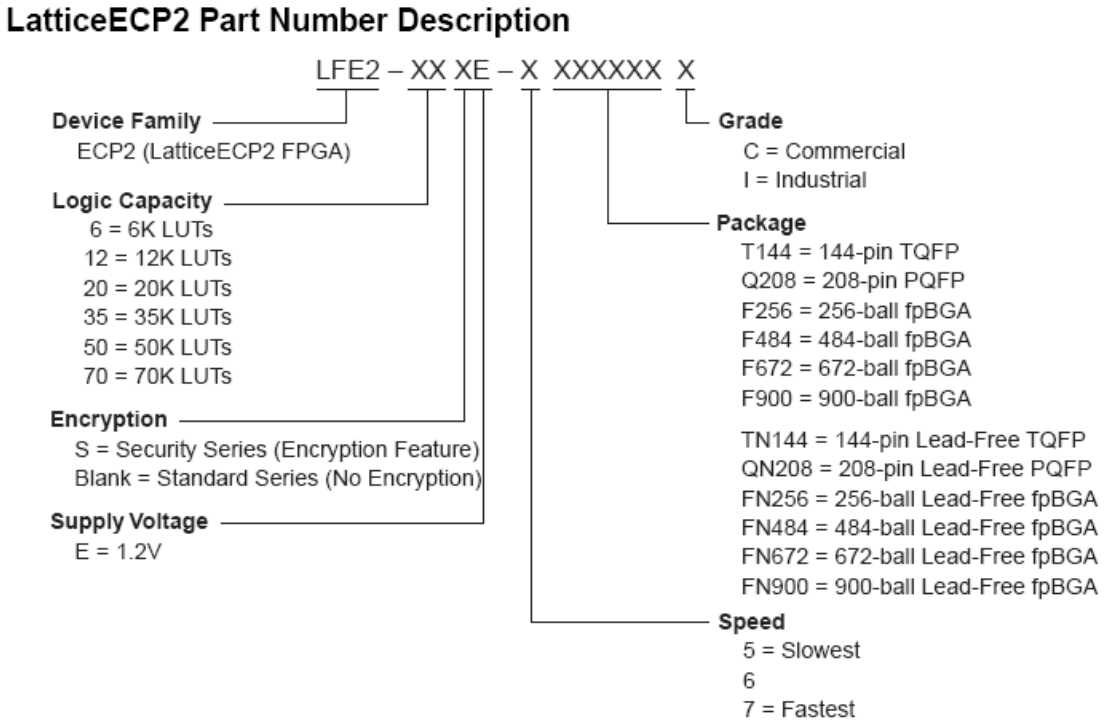


Figura III.2

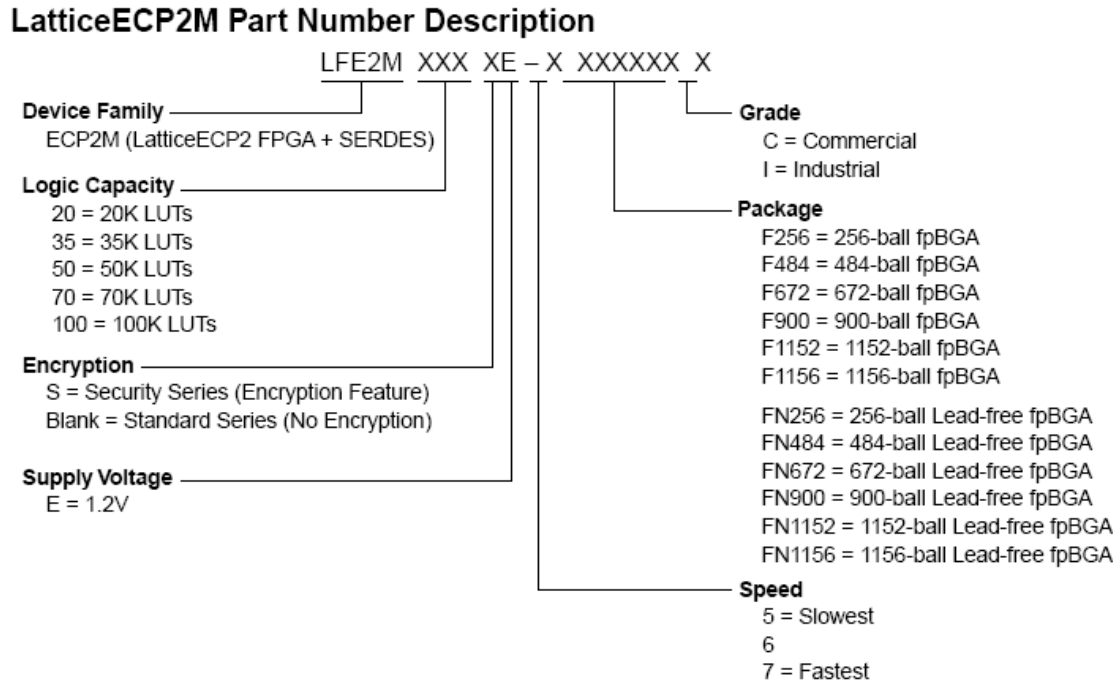


Figura III.3

III.2.3 Familias de dispositivos LatticeECP & EC

Para revisar una información más detallada sobre estos dispositivos, consulte el apartado V.3 DISPOSITIVOS LATTICE ECP & EC.

III.2.3.1 Datos generales

La familia de dispositivos LatticeECP/EC está optimizada para mantener las principales características de las FPGAs y reducir el coste. La familia LatticeECP (EConomy Plus) combina como base una FPGA eficiente a la que se han añadido funciones dedicadas de alta velocidad. La familia LatticeEC (EConomy) presenta las características de propósito general de los dispositivos LatticeECP, pero sin los bloques de funciones dedicadas, para conseguir soluciones de menor coste.

La familia de FPGAs de Lattice ECP/EC utiliza la tecnología de lógica reconfigurable SRAM, lógica basada en LUT (lookup table). Los dispositivos de esta familia tienen desde 1,5K hasta 32,8K LUTs, además presentan entre 65 y 496 pines de entrada/salida de usuario, contienen bloques de memoria interna sysMEM, PLLs, entre 0 y 32 multiplicadores de 18x18, entradas/salidas paralelas estándar y entradas/salidas serie.

Los dispositivos de esta familia ofrecen entre 0 y 8 sysDSPs, Procesadores Digitales de Señales, diseñados expresamente para conseguir un alto rendimiento en el procesado digital de señales, a bajo coste.

Los dispositivos de la familia ECP/EC tienen un puerto sysCONFIG que permite la configuración tanto serie como paralelo del dispositivo.

III.2.3.2 Configuración

Todos los dispositivos LatticeECP/EC tienen dos puertos que pueden utilizarse para la configuración del dispositivo. El Test Access Port (TAP) soporta configuración bit-wide y el puerto sysCONFIG soporta configuración byte-wide y serie. El puerto sysCONFIG es un interfaz de 20 pines con seis entradas/salidas utilizadas como pines dedicados y el resto como pines de doble uso.

En el momento del encendido la memoria SRAM de la FPGA está preparada para ser configurada a través del puerto sysCONFIG. El puerto TAP se puede activar en cualquier momento tras el arranque, enviando el comando apropiado a través del puerto TAP. Una vez seleccionado un puerto de configuración, éste permanece seleccionado de tal modo que no se podrá seleccionar otro puerto de configuración hasta la siguiente secuencia de arranque.

Además, cada dispositivo de la familia tiene un puerto JTAG.

III.2.3.3 Dispositivos

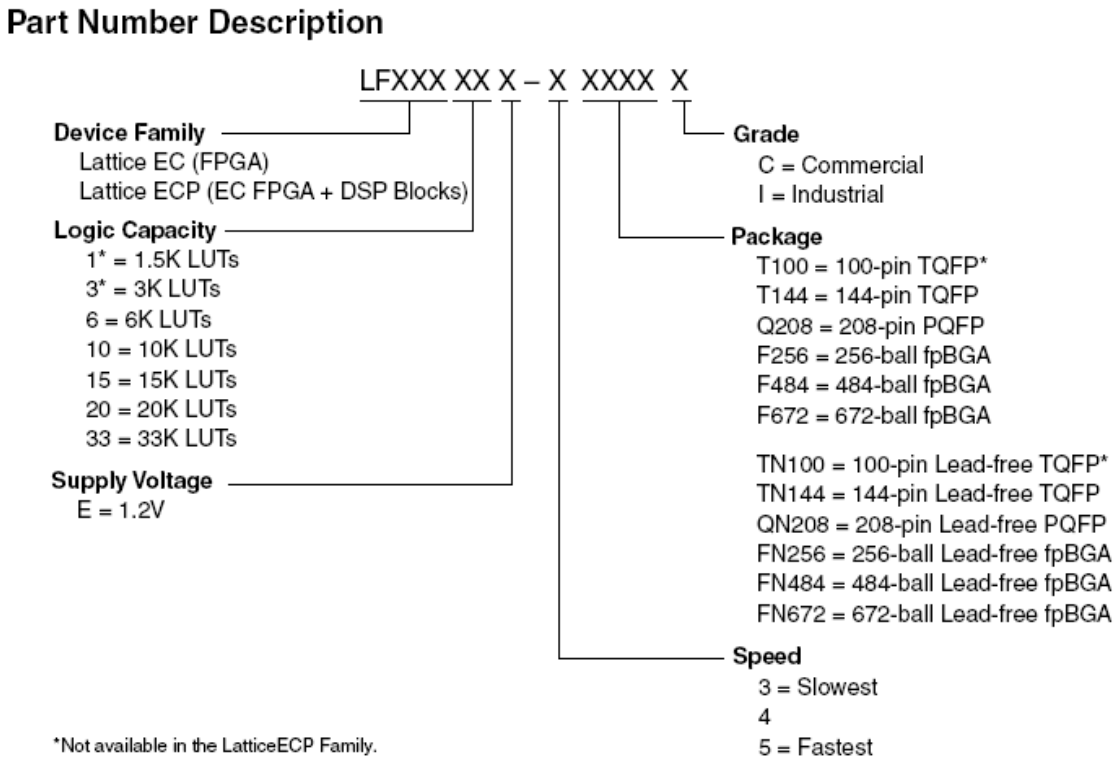


Figura III.4

III.2.4 Familias de dispositivos LatticeSC & LatticeSCM

Para revisar una información más detallada sobre estos dispositivos, consulte el apartado V.4 DISPOSITIVOS LATTICE SC & LATTICE SCM.

III.2.4.1 Datos generales

La familia de FPGAs LatticeSC combina una FPGA de alto rendimiento, SERDES de alta velocidad (de 600Mbps a 3.8Gbps), entradas/salidas paralelo de alto rendimiento (2Gbps) y gran cantidad de bloques de memoria RAM embebida.

La familia de FPGAs LatticeSC utiliza la tecnología de lógica reconfigurable SRAM, lógica basada en LUT (lookup table). Los dispositivos de esta familia tienen desde 15K hasta 115K LUTs, además presentan entre 139 y 942 pines de entrada/salida de usuario, contienen bloques de memoria interna sysMEM, PLLs, DLLs, entradas/salidas paralelas estándar y entradas/salidas serie.

La familia de dispositivos Lattice SC está optimizada para ofrecer características de alto rendimiento para conseguir interfaces de alta velocidad, con vistas a la interconexión de sistemas, permitiendo la interconexión de sistemas utilizando estándares como Ethernet, PCI Express y SPI4.2.

Los dispositivos de la familia SC tienen un puerto sysCONFIG que permite la configuración tanto serie como paralelo del dispositivo.

III.2.4.2 Configuración

Todos los dispositivos LatticeSC tienen 3 puertos que pueden ser usados para la programación del dispositivo. El puerto serie, que soporta la configuración bit-wide, el puerto sysCONFIG, que soporta la configuración byte-wide y serie, y el puerto MPI que soporta configuraciones de 8-bit, 16-bit y 32-bit.

El puerto sysCONFIG es un interfaz de 20 pines con seis entradas/salidas utilizadas como pines dedicados y el resto como pines de doble uso. Cuando no se utiliza el modo sysCONFIG esos pines de doble uso están disponibles como entradas/salidas de propósito general.

En el arranque, la memoria SRAM de la FPGA está preparada para ser configurada por el puerto sysCONFIG. El puerto TAP se puede activar en cualquier momento tras el arranque, enviando el comando apropiado a través del puerto TAP. Una vez que se selecciona un puerto de configuración este puerto permanece seleccionado y los demás puertos de configuración no pueden ser activados hasta la siguiente secuencia de reinicio.

Además, todos los dispositivos de esta familia tienen un puerto JTAG.

III.2.4.3 Dispositivos

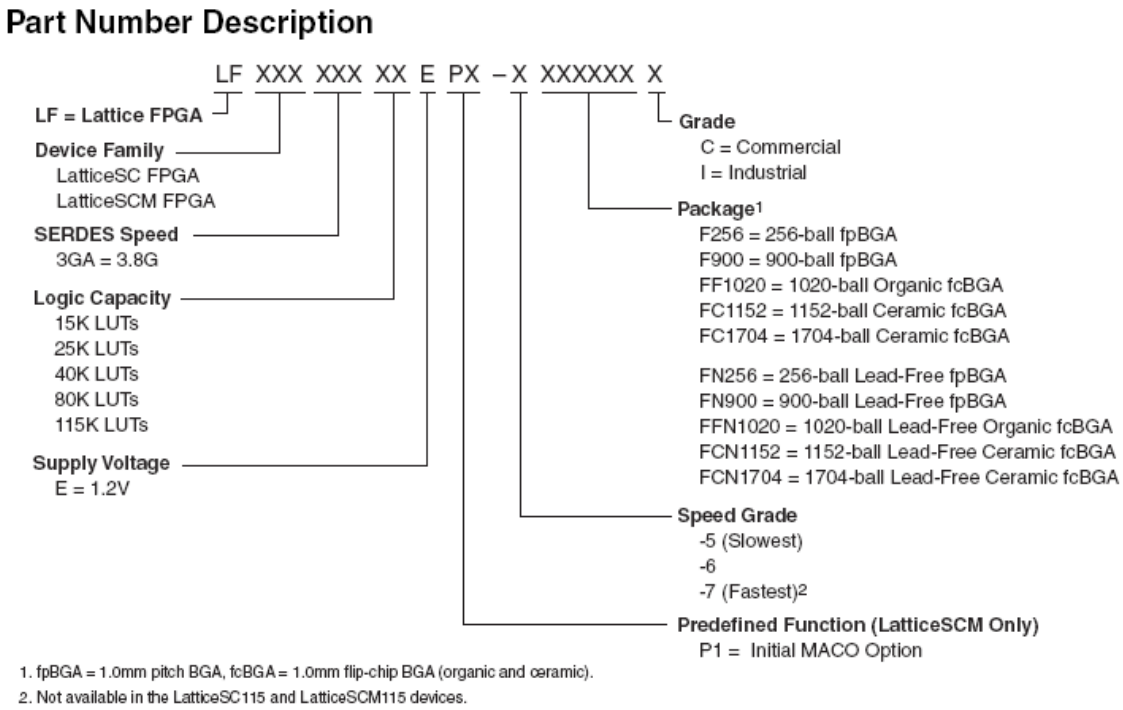


Figura III.5

III.2.5 Familia de dispositivos LatticeXP2

Para revisar una información más detallada sobre estos dispositivos, consulte el apartado V.5 DISPOSITIVOS LATTICE XP2.

III.2.5.1 Datos generales

Los dispositivos Lattice XP2 combinan una estructura FPGA basada en LUTs con celdas FLASH no volátiles en una arquitectura llamada flexiFLASH. La presencia de bloques flexiFLASH proporciona encendido instantáneo, reconfigurabilidad indefinida y almacenamiento en el propio chip.

Los dispositivos de esta familia tienen desde 5K hasta 40K LUTs, además presentan entre 86 y 540 pines de entrada/salida de usuario, contienen bloques de memoria interna sysMEM, PLLs, entre 12 y 32 multiplicadores de 18x18, entradas/salidas paralelas estándar y entradas/salidas serie.

Los dispositivos de esta familia ofrecen entre 3 y 8 sysDSPs, Procesadores Digitales de Señales, diseñados expresamente para conseguir un alto rendimiento en el procesado digital de señales, a bajo coste.

Todos los dispositivos de la familia LatticeXP2 presentan un puerto sysCONFIG multiplexado con uno de los 8 bancos de entradas/salidas (sysIO), que soporta la configuración serie y paralelo del dispositivo. Los dispositivos también presentan un puerto JTAG.

III.2.5.2 Configuración

Los dispositivos LatticeXP2 combinan, en un solo chip, memorias FLASH y SRAM para ofrecer flexibilidad en la programación y configuración.

En el arranque, o por comando de usuario, los datos se transfieren desde la memoria FLASH del chip a las celdas SRAM de configuración que controlan las operaciones del dispositivo. Estos datos se transfieren por los buses de forma masiva, en paralelo; quedando los dispositivos preparados para su funcionamiento en microsegundos, permitiéndose así el arranque instantáneo.

La presencia de memoria FLASH en el chip elimina la necesidad de memoria externa para el arranque. Esta memoria FLASH puede ser configurada a través de ambos puertos, el JTAG y el SPI esclavo. Las celdas SRAM pueden ser indefinidamente reconfiguradas a través de los puertos JTAG o el SPI maestro del dispositivo.

III.2.5.3 Dispositivos

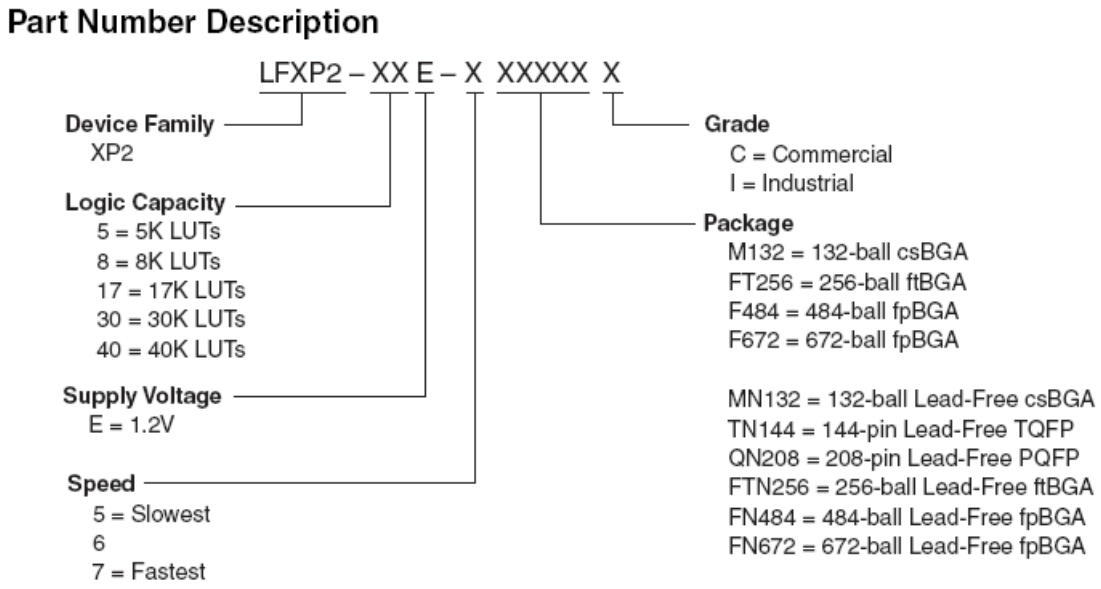


Figura III.6

III.2.6 Familia de dispositivos LatticeXP

Para revisar una información más detallada sobre estos dispositivos, consulte el apartado V.6 DISPOSITIVOS LATTICE XP.

III.2.6.1 Datos generales

La familia de dispositivos Lattice XP combina, en una arquitectura no volátil e indefinidamente reconfigurable, puertas lógicas, memoria embebida y entradas/salidas de alto rendimiento.

Los dispositivos de esta familia tienen desde 3K hasta 20K LUTs, además presentan entre 62 y 340 pines de entrada/salida de usuario, contienen bloques de memoria interna sysMEM, PLLs, entradas/salidas paralelas estándar y entradas/salidas serie.

Todos los dispositivos de la familia Lattice XP presentan un puerto sysCONFIG, que soporta la configuración serie y paralelo del dispositivo. Los dispositivos también presentan un puerto JTAG.

III.2.6.2 Configuración

Todos los dispositivos LatticeXP tienen la posibilidad de configurarse y programarse a través de dos puertos. El Puerto de Acceso de Test (TAP) soporta la configuración serie y el puerto sysCONFIG soporta la configuración byte-wide y serie.

En el arranque, o por comando de usuario, los datos se transfieren desde la memoria FLASH del chip a las celdas SRAM del dispositivo; consiguiéndose de este modo el arranque instantáneo del dispositivo.

La memoria SRAM puede ser indefinidamente reconfigurada a través del puerto JTAG, del puerto TAP, del puerto sysCONFIG y, como se explicó en el párrafo anterior, utilizando los datos almacenados en la memoria FLASH del propio chip.

La memoria FLASH puede ser configurada a través de los puertos JTAG y sysCONFIG.

III.2.6.3 Dispositivos

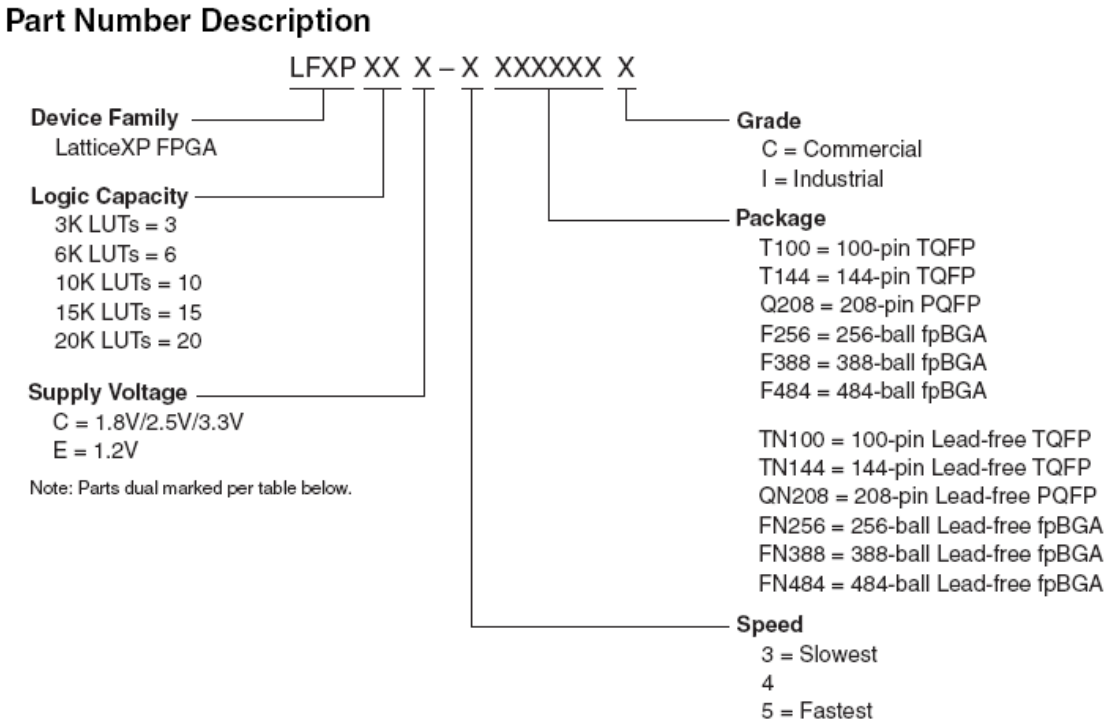


Figura III.7

III.2.7 Familia de dispositivos ispXPGA

Para revisar una información más detallada sobre estos dispositivos, consulte el apartado V.7 DISPOSITIVOS ISPXPGA.

III.2.7.1 Datos generales

Todos los dispositivos de Lattice pertenecientes a la familia ispXPGA son dispositivos no volátiles e indefinidamente reconfigurables.

La familia ispXPGA está disponible en dos opciones, los dispositivos estándar soportan sysHSI, que es la capacidad para comunicaciones serie hasta 800Mbps; mientras que las “series-E” presentan las mismas características FPGA sin el bloque sysHSI. El bloque sysHSI es el precursor de los bloques SERDES de alta velocidad, que incluyen algunas de las familias ya estudiadas, ya que estos bloque sysHSI ya realizan la serialización/de-serialización.

Las celdas de memoria CMOS borrable eléctricamente (EECMOS) proporcionan a la familia ispXPGA la no volatilidad. Esto permite que, tras el encendido, la lógica esté operativa en cuestión de microsegundos. Las celdas SRAM internas permiten que el dispositivo sea reconfigurado indefinidas veces. Estas celdas SRAM pueden ser configuradas y leídas a través del puerto sysCONFIG.

La familia ofrece entre 139K y 1,25M de puertas lógicas y entre 160 y 496 pines de entrada/salida, contienen bloques de memoria interna sysMEM, PLLs, entradas/salidas paralelas estándar y entradas/salidas serie.

Los dispositivos de la familia ispXPGA pueden ser configurados en modo serie a través del puerto TAP (Test Access Port) y en modo paralelo usando el puerto sysCONFIG.

III.2.7.2 Configuración

La familia de dispositivos ispXPGA presenta dos tipos de memoria, RAM estática y EECMOS no volátil. La RAM estática se utiliza para el control de la funcionalidad del dispositivo durante la operación normal del mismo y la EECMOS se utiliza para cargar la SRAM. Hay una relación de uno a uno entre la memoria SRAM y la EECMOS. La memoria SRAM puede ser configurada desde la memoria EECMOS o desde una fuente externa.

Hay dos puertos desde los cuales se puede configurar la memoria SRAM: El puerto TAP para configuraciones bit-wide y el puerto sysCONFIG que permite configuraciones byte-wide. Para la programación de la memoria EECMOS solo está disponible el puerto TAP.

Tras el encendido la memoria SRAM del dispositivo puede ser configurada desde la memoria EECMOS o desde una fuente externa a través del modo sysCONFIG. Además la memoria SRAM puede ser reconfigurada desde la memoria EECMOS ejecutando un refresco.

III.2.7.3 Dispositivos

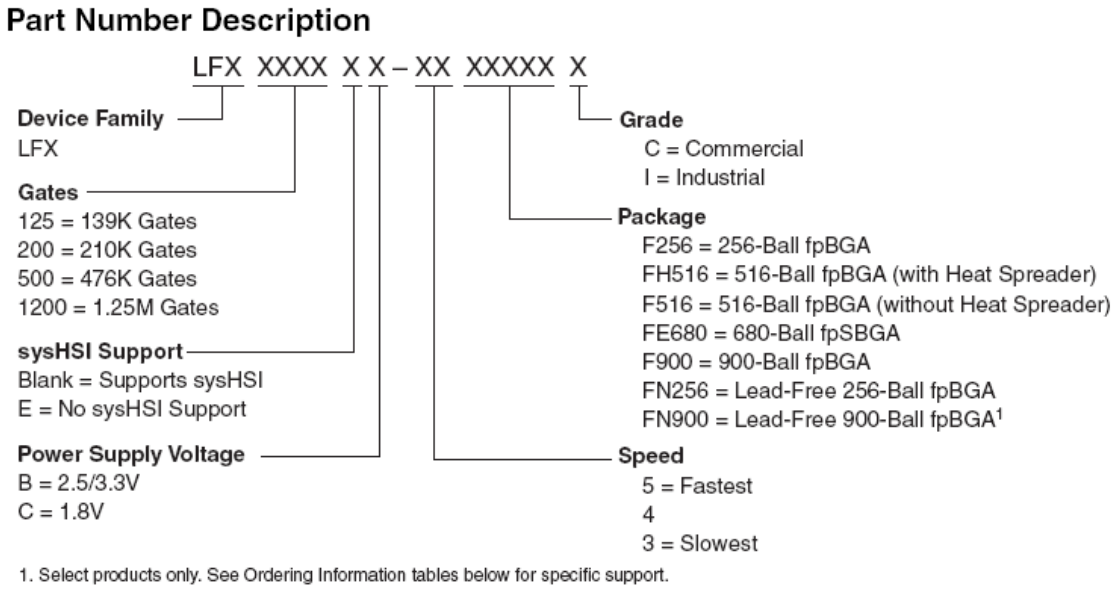


Figura III.8

III.3 PLDs

III.3.1 CPLDs

III.3.1.1 Familia de dispositivos MachXO2

Para revisar una información más detallada sobre estos dispositivos, consulte el apartado V.8 DISPOSITIVOS MACHXO2.

III.3.1.1.1 Datos generales

La familia de dispositivos de Lattice MachXO2 utiliza la tecnología de lógica reconfigurable SRAM, lógica basada en LUT (lookup table). Los dispositivos de esta familia tienen desde 256 hasta 6864 LUTs, además presentan entre 19 y 335 pines de entrada/salida de usuario, contienen bloques de memoria interna sysMEM, PLLs, entradas/salidas paralelas estándar y entradas/salidas serie.

Esta familia de dispositivos es la evolución de la familia MachXO, que será la siguiente en ser expuesta. En esta evolución se ha ampliado el número de LUTs, aunque este número sigue siendo muy pequeño en comparación con la cantidad de LUTs que ofrecen las familias de FPGAs. Además incluye bloques dedicados a la entrada/salida síncrona y también añade en el propio dispositivo memoria FLASH que puede ser reescrita hasta 100.000 veces.

Hay dos puertos desde los cuales se pueden configurar los dispositivos de esta familia: El puerto TAP para configuraciones bit-wide y el puerto sysCONFIG que permite configuraciones byte-wide. Para la programación

de la memoria FLASH solo está disponible el interfaz WISHBONE que está incluido en el dispositivo con este propósito.

III.3.1.1.2 Configuración

Todos los dispositivos MachXO2 contienen dos puertos que pueden ser utilizados para la configuración del dispositivo. Como se indicó en el apartado anterior, el puerto TAP permite la configuración bit-wide y el puerto sysCONFIG que permite la configuración serie.

Al encenderse los dispositivos tienen el puerto sysCONFIG habilitado para la configuración, pudiendo activarse el puerto TAP mediante el envío del comando apropiado por el mismo puerto (TAP). Salvo en el encendido, cuando se selecciona un puerto, éste permanece activo hasta que termina el ciclo de configuración.

III.3.1.1.3 Dispositivos

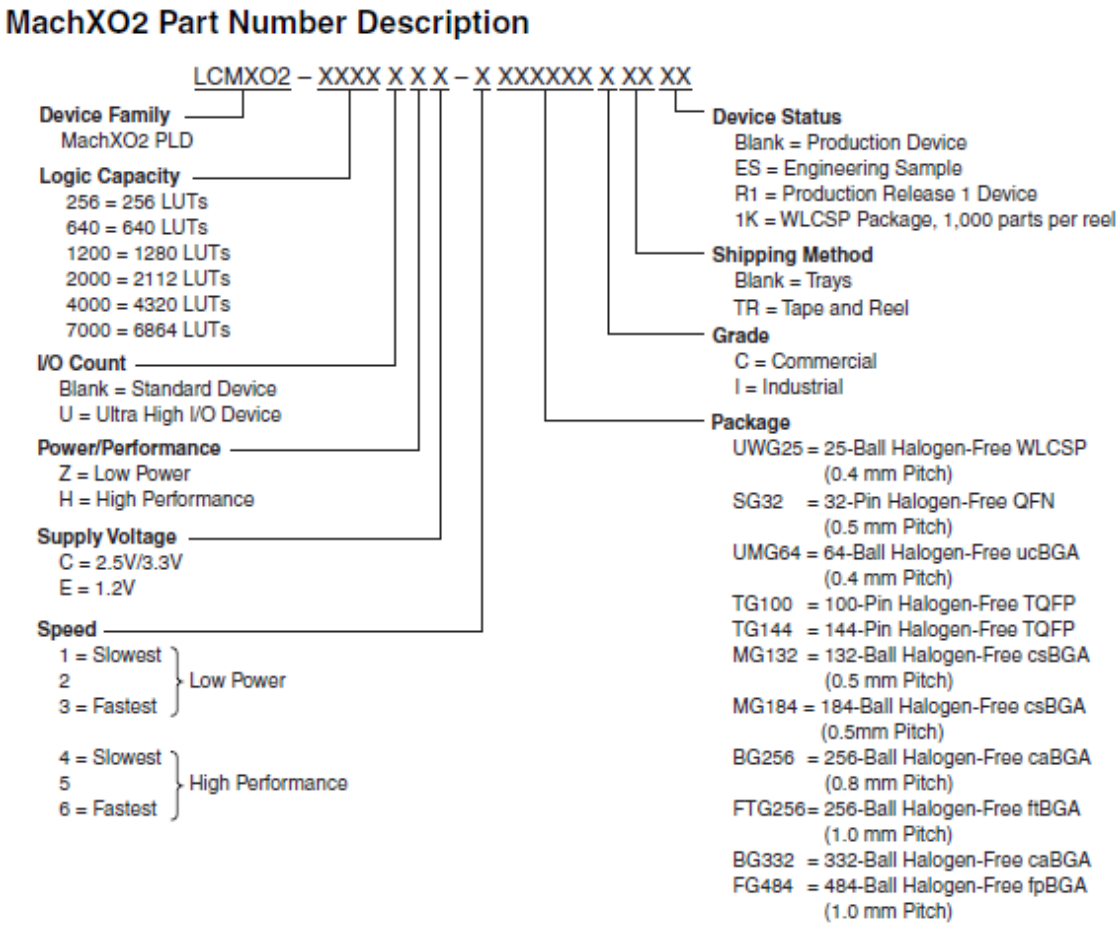


Figura III.9

III.3.1.2 Familia de dispositivos MachXO

Para revisar una información más detallada sobre estos dispositivos, consulte el apartado V.9 DISPOSITIVOS MACHXO.

III.3.1.2.1 Datos generales

La familia de dispositivos de Lattice MachXO está optimizada para adaptarse a las aplicaciones que tradicionalmente se llevaban a cabo con CPLDs y FPGAs de baja capacidad, como son: pequeñas cantidades de lógica, interfaces de buses, control de arranque y control lógico.

Los dispositivos Lattice MachXO utilizan LUTs y bloques de memoria embebida, que tradicionalmente se asocian a las FPGAs, para tener flexibilidad e implementar lógica eficiente. A través de la tecnología no volátil los dispositivos proporcionan soluciones en un único chip, de alta seguridad y con arranque instantáneo, estas capacidades se suelen asociar con los CPLDs.

La familia de dispositivos de Lattice MachXO utiliza la tecnología de lógica reconfigurable SRAM, lógica basada en LUT (lookup table). Los dispositivos de esta familia tienen desde 256 hasta 2280 LUTs, además presentan entre 73 y 271 pines de entrada/salida de usuario, contienen bloques de memoria interna sysMEM, hasta 2 PLLs, buffers de entrada/salida programables para soportar algunos interfaces estándar.

Cada dispositivo de la familia tiene un puerto JTAG que soporta la configuración y programación del dispositivo y el acceso a la lógica de usuario.

III.3.1.2.2 Configuración

Todos los dispositivos MachXO contienen un puerto de acceso de test que puede ser utilizado para la configuración y programación del dispositivo.

La memoria SRAM puede configurarse desde la memoria no volátil del propio chip, durante el encendido del dispositivo y utilizando el puerto TAP.

III.3.1.2.3 Dispositivos

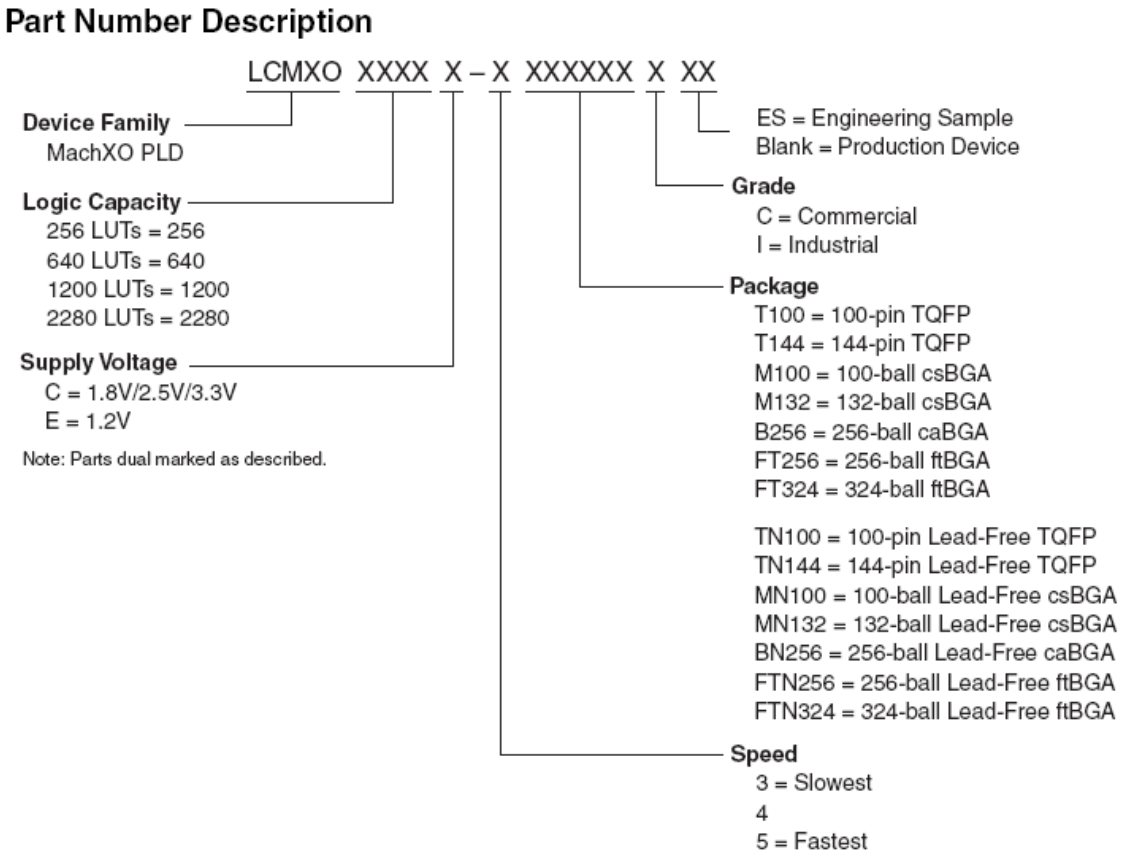


Figura III.10

III.3.1.3 Dispositivos CPLD ispMACH 4000ZE

Para revisar una información más detallada sobre estos dispositivos, consulte el apartado V.10 DISPOSITIVOS ISPMACH 4000ZE.

III.3.1.3.1 Datos generales

La familia de dispositivos de alto rendimiento de Lattice ispMACH 4000ZE está basada en la arquitectura ispMACH 4000, mejorándola en cuanto a rendimiento y consumo.

La familia ispMACH 4000ZE ofrece densidades de integración de entre 32 y 256 macrocells. Hay varias combinaciones de densidades de entradas/salidas entre 32 y 144 pines/bolas.

En el dispositivo se incluye un oscilador interno y un timer para tareas como control de LEDs y escaneo de teclados. Estas capacidades pueden desactivarse para ahorrar energía.

La familia de dispositivos de Lattice ispMACH 4000ZE solo soporta fuentes de alimentación de 1,8V y opera con interfaces de voltaje de 3,3V 2,5V, 1,8V y 1,2V. Además la familia es tolerante a interfaces de 5V.

La familia de dispositivos ispMACH 4000ZE también ofrece características de entrada/salida mejoradas, tales como control del slew rate, resistencias de pull-up y pull-down y salidas en drenador abierto.

Los miembros de la familia ispMACH 4000ZE son dispositivos programables in-system.

III.3.1.3.2 Configuración

Todos los dispositivos de la familia ispMACH 4000ZE presentan la capacidad de ser programados in-system (capacidad ISP) a través del puerto de acceso de test (TAP).

III.3.1.3.3 Dispositivos

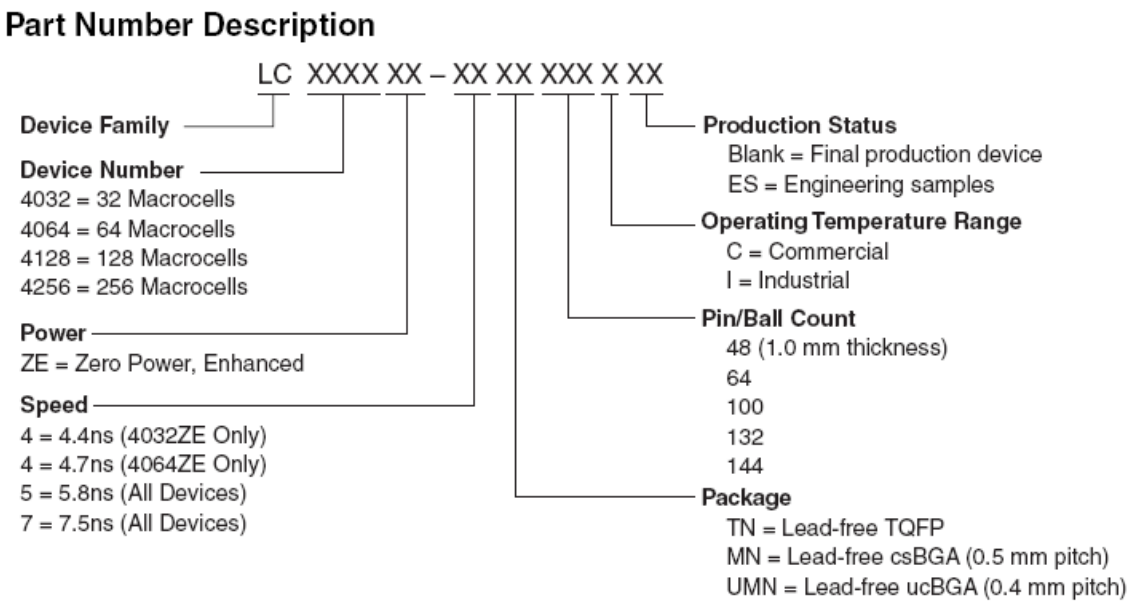


Figura III.11

III.3.1.4 Línea principal de CPLD ispMACH 4000V/B/C/Z

Para revisar una información más detallada sobre estos dispositivos, consulte el apartado V.11 DISPOSITIVOS ISPMACH 4000V/B/C/Z.

III.3.1.4.1 Datos generales

La familia de dispositivos de Lattice ispMACH 4000 ofrece una solución CPLD con densidades de integración de entre 32 y 512 macrocells. Hay varias combinaciones de densidades de entradas/salidas entre 44 y 256 pines/bolas.

La familia ispMACH 4000 soporta fuentes de alimentación de 3,3V, 2,5V y 1,8V e interfaces de 3,3V, 2,5V y 1,8V. Además es tolerante a interfaces de 5V.

La familia de dispositivos ispMACH 4000 también ofrece características de entrada/salida mejoradas, tales como control del slew rate, resistencias de pull-up y pull-down y salidas en drenador abierto.

Los miembros de la familia ispMACH 4000 son dispositivos de 3,3V, 2,5V y 1,8V programables in-system.

III.3.1.4.2 Configuración

Todos los dispositivos de la familia ispMACH 4000 presentan la capacidad de ser programados in-system (capacidad ISP) a través del puerto de acceso de test (TAP).

III.3.1.4.3 Dispositivos

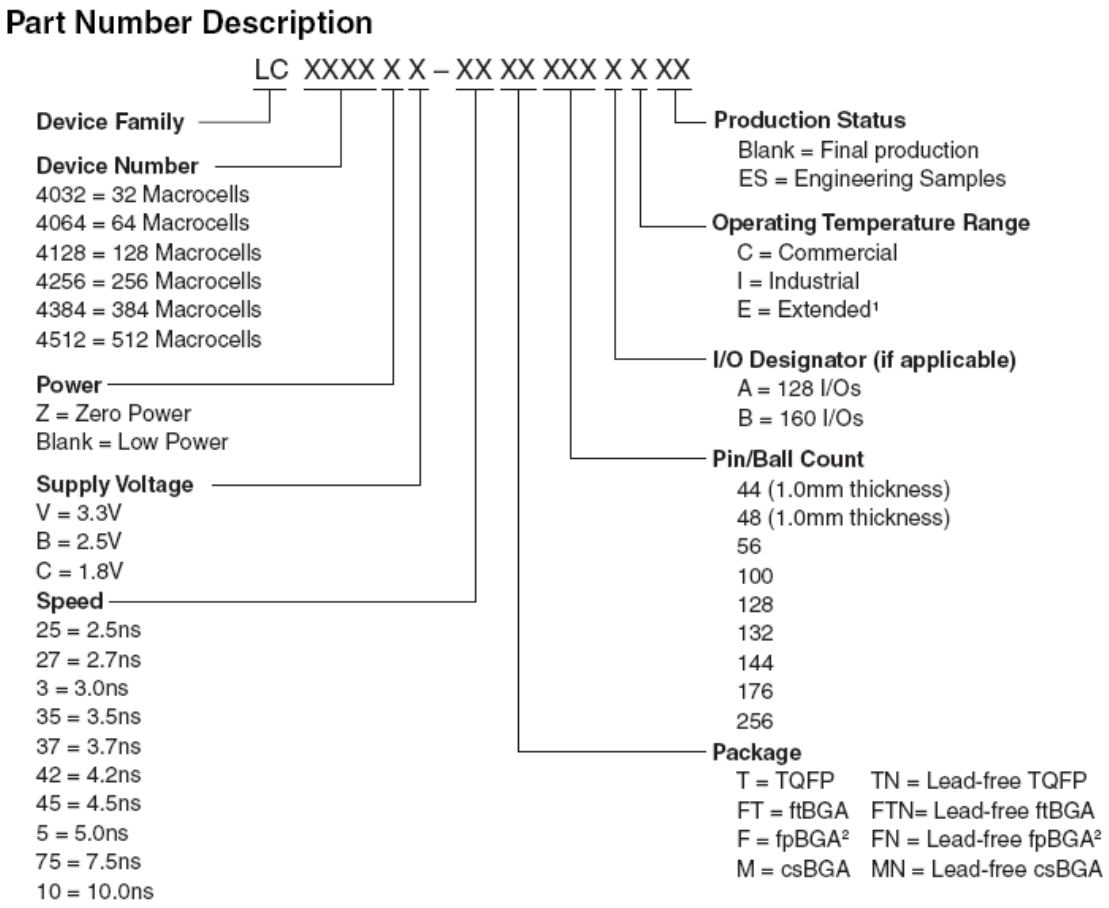


Figura III.12

III.3.2 SPLDs

III.3.2.1 Dispositivos PLD simples GAL y dispositivos GAL programables In-System ispGAL

III.3.2.1.1 Datos generales

Lattice ofrece varias opciones de arquitectura, tensión, potencia y rendimiento para sus productos de GAL. Estos productos GAL son adecuados para añadir cantidades de lógica muy pequeñas.

La familia ispGAL permite la programación in-system (ISP) aunque no en todos sus miembros. Está disponible en 5V, 3,3V, 2,5V y 1,8V. Pueden funcionar a frecuencias de hasta 455MHz.

Los dispositivos de estas familias contienen un número de salidas programables que varía entre 8 y 12 en función del dispositivo elegido. Cada una de esas salidas se corresponde a un elemento de la arquitectura llamado Macrocelda Lógica de Salida, OLMC por sus siglas en inglés (Output Logic Macrocell). Cada OLMC presenta un producto con un número variable de términos, el número de términos de cada producto dentro de un mismo dispositivo es fijo pero varía de un dispositivo a otro.

III.3.2.1.2 Configuración

La programación completa del dispositivo tarda solo unos segundos. El borrado del dispositivo es transparente para el usuario, se realiza automáticamente como parte del ciclo de programación. La familia ispGAL ofrece todos los beneficios de la programación in-system (ISP).

III.3.2.1.3 Dispositivos

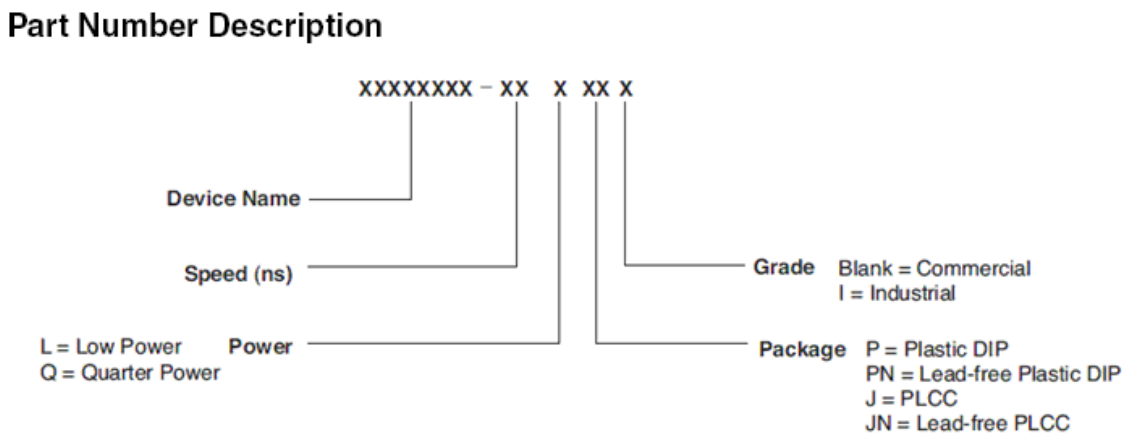


Figura III.13

III.4 CONCLUSIONES RESPECTO A QUÉ DISPOSITIVOS UTILIZAR EN UNA PLACA DE PROTOTIPADO PARA LA REALIZACIÓN DE PRÁCTICAS EN LABORATORIO

Hemos de tener en cuenta que las prácticas de laboratorio han de partir desde cero y alcanzarán un grado de complejidad medio. Razón por la cual, necesitaremos dispositivos que no sean de muy baja capacidad, pero tampoco sería bueno que fuesen de alta capacidad; para que el alumnado tenga que ceñirse a unos límites en este aspecto. De este modo, también ajustamos los costes de fabricación de las tarjetas que serán utilizadas en el laboratorio.

Dichas tarjetas presentarán, entre otros, cuatro displays BCD-7Segmentos, 2 bloques de 8 interruptores, 16 pulsadores para simular un teclado hexadecimal, otros 4 pulsadores para otras funciones como por ejemplo el Reset, puerto para comunicaciones serie, puerto para comunicaciones paralelo y puerto específico para la programación del dispositivo elegido.

Por lo tanto, el dispositivo que se utilizará tiene que ser capaz de gestionar y conectarse a todo lo anteriormente descrito y en caso de presentar otras funcionalidades no serían utilizadas.

Además, como cada tarjeta será configurada un gran número de veces es necesario que el dispositivo elegido sea indefinidamente reconfigurable.

III.4.1 Familia de dispositivos LatticeECP3

La familia de FPGAs de Lattice ECP3 presenta funcionalidades de alto rendimiento que no serían utilizadas. Por lo tanto se descartan los dispositivos de esta familia por su excesiva capacidad y potencia funcional.

III.4.2 Familia de dispositivos LatticeECP2/M

La familia de dispositivos LatticeECP2/M también presenta funcionalidades de alto rendimiento que no serían utilizadas y supondrían un incremento del coste no justificado por la obtención de ningún beneficio.

III.4.3 Familias de dispositivos LatticeECP & EC

La familia de dispositivos LatticeECP/EC presenta dos variantes, la ECP que presenta funciones dedicadas de alta velocidad, que no serían utilizadas. Y por lo tanto, queda descartada. La otra variante, la EC, presenta las características FPGA optimizadas como la variante ECP y además carece de las funciones de alta velocidad, razón por la cual los dispositivos de esta familia y en concreto los de la variante EC son adecuados para su inclusión en la tarjeta.

III.4.4 Familias de dispositivos LatticeSC & LatticeSCM

La familia de FPGAs de alto rendimiento LatticeSC presenta capacidad SERDES de alta velocidad, entradas/salidas de alto rendimiento y gran memoria RAM embebida. Y todo ello no será utilizado, con lo cual esta familia queda descartada.

III.4.5 Familia de dispositivos LatticeXP2

Los dispositivos LatticeXP2 son una evolución de los dispositivos XP a los cuales se han añadido funcionalidades como el encendido instantáneo y la memoria en el propio chip que no serán utilizadas y por lo tanto descartamos esta familia de dispositivos por comparación con la familia XP.

III.4.6 Familia de dispositivos LatticeXP

La familia de dispositivos LatticeXP presentan unas características ajustadas a lo que será necesario sin presentar exceso de capacidades consiguiendo de este modo un coste ajustado.

III.4.7 Familia de dispositivos ispXPGA

La familia ispXPGA presenta funcionalidades de alto rendimiento que no serían utilizadas y supondrían un incremento del coste no justificado por la obtención de ningún beneficio.

III.4.8 Familia de dispositivos MachXO2

La familia de dispositivos MachXO2, aún siendo una evolución de la familia MachXO, continúa ofreciendo muy poca capacidad (6864 LUTs) para realizar los diseños que serán exigidos. Se descarta por falta de capacidad.

III.4.9 Familia de dispositivos MachXO

La familia de dispositivos de Lattice MachXO presenta un máximo de 2280 LUTs, que es escaso para realizar los diseños de mediana complejidad que serán exigidos. Por lo tanto esta familia de dispositivos queda descartada por falta de capacidad.

III.4.10 Dispositivos CPLD ispMACH 4000ZE

La familia de dispositivos de Lattice ispMACH 4000ZE ofrece un máximo de 256 macrocells que son insuficientes para realizar los diseños de mediana complejidad que serán exigidos. Por lo tanto esta familia de dispositivos queda descartada por falta de capacidad. Además es una evolución, mejorada principalmente desde el punto de vista del consumo de energía, de la familia ispMACH 4000V/B/C/Z, con lo cual presentaría mejoras que no son necesarias y por lo tanto que no justificarían el incremento del coste.

III.4.11 Línea principal de CPLD ispMACH 4000V/B/C/Z

La familia de dispositivos de Lattice ispMACH 4000V/B/C/Z ofrece un máximo de 512 macrocells que son insuficientes para realizar los diseños de mediana complejidad que serán exigidos. Por lo tanto esta familia de dispositivos queda descartada por falta de capacidad.

III.4.12 Dispositivos PLD simples GAL y dispositivos GAL programables In-System ispGAL

Las familias de dispositivos PLD simples GAL y dispositivos GAL programables In-System ispGAL de Lattice son adecuados para añadir pequeñas cantidades de lógica, o para proporcionar una rápida solución a un problema crítico de la lógica del sistema. Razón por la cual resultan inadecuados para realizar los diseños de mediana complejidad que serán exigidos. Por lo tanto esta familia de dispositivos queda descartada por falta de capacidad.

IV Capítulo IV: “Conclusiones”

IV.1 CONCLUSIONES RESPECTO A LA VIABILIDAD DEL ENTORNO PARA LA ENSEÑANZA DE HERRAMIENTAS DE CAD

Aunque a lo largo del documento ya se han ido comentando algunas de las conclusiones a las que se fue llegando, se ha querido incorporar este capítulo para reunir y desarrollar brevemente todas aquellas conclusiones acerca de la viabilidad del entorno para la enseñanza de CAD electrónico.

Los puntos a favor de decantarnos por utilizar este entorno para la enseñanza de herramientas de CAD son:

El hecho de que para la captura de esquemas, el entorno ispLEVER, permite realizar tanto esquemáticos planos como jerárquicos, en los que los diseños están definidos en más de un nivel. Esto permite que el alumno entienda, desde un principio, que los diseños son algo escalable y que los diseños grandes pueden ir siendo divididos en bloques cada vez más pequeños para abordar su desarrollo. Incluso podría plantearse que la última práctica de un laboratorio consistiese en la creación de un diseño de gran complejidad, en el que cada alumno tuviese que realizar un bloque distinto. Así se reforzaría la divisibilidad de los diseños, a la vez que se desarrollaría el trabajo en equipo, al tener que ponerse de acuerdo para que los módulos de unos y otros alumnos se puedan comunicar entre sí.

Que el editor de esquemáticos se combine con el navegador jerárquico, el editor de símbolos y las librerías de símbolos para permitir la revisión y la modificación de los diseños de una manera cómoda para el alumno, ya que todas ellas son herramientas lo suficientemente intuitivas, como para que su aprendizaje no necesite más que una única utilización guiada, para conseguir entender su funcionamiento. El navegador jerárquico ayuda a mantener en orden los distintos niveles dentro de un diseño, reforzando los beneficios de la escalabilidad de los diseños.

Que, como en general el resto del entorno, la captura de esquemas resulte fácil. Esto permite que no solo la captura inicial sea sencilla, sino que también los cambios se realicen de forma rápida; permitiendo al alumno depurar los diseños ágilmente.

La posibilidad de ajustar el tamaño de la hoja al del diseño hace que los diseños estén más organizados y facilita la visualización; al no tener que estar continuamente utilizando el zoom para poder ver diseños o bloques pequeños con suficiente nitidez.

Que es un entorno que, en líneas generales, resulta fácil de manejar. Si el entorno no fuese sencillo y fácil, podría generar en el alumno un sentimiento de rechazo; este rechazo obstaculizaría el proceso de aprendizaje ya que podría desmotivar al alumno. En cambio, ser un entorno sencillo y fácil predispone al alumno a dedicarle tiempo.

Que exista la posibilidad de crear símbolos de los diseños realizados por el usuario, y añadirlos como parte de diseños posteriores. Esta posibilidad afianza en el alumno la idea de la escalabilidad de los diseños, ya que el

símbolo de lo que hoy es un diseño completo mañana lo puede utilizar como parte de un diseño mayor.

Que la especificación de diseños permita la captura y editores para VHDL y Verilog. Esto permitiría hacer llegar a los alumnos ejemplos de los diseños que han de realizar, definidos utilizando los editores para que los alumnos puedan realizar simulaciones pero no copiar los esquemas. De este modo pueden observar cómo ha de comportarse el diseño y compararlo con los resultados de sus simulaciones. Podría incluso, crearse alguna práctica que se centrara en las simulaciones, para fomentar el uso de éstas. Ya que debería emplearse más tiempo en realizar simulaciones que en capturar esquemas y el tiempo que los alumnos invierten en un diseño suele estar dividido en un 80% realizando la captura y solo un 20% realizando simulaciones, siendo contrario a la práctica correcta.

Que el entorno permite realizar ciclos completos, llegando a programar dispositivos físicos del fabricante. Gracias a esto se puede enseñar al alumno todo el proceso, desde la recepción de las especificaciones como enunciado hasta la entrega del diseño ya materializado en un dispositivo.

El gran nivel de integración entre las distintas herramientas del entorno, no presentándose la necesidad de ejecutar herramientas desde fuera del entorno. Todo el entorno está integrado de tal manera que para el usuario resulta natural el cambio de una herramienta a otra.

Que el fabricante permite la descarga de una versión de evaluación del entorno. Esto permite al alumno tener una versión en su casa para poder reforzar el estudio y avanzar trabajos sin la necesidad de utilizar un puesto de laboratorio.

Los puntos en contra son:

El entorno ofrece varios métodos para definir las formas de onda usadas en las simulaciones, ya que es posible importar archivos que contengan los vectores de test, definir los estímulos de forma textual o definir los estímulos de forma gráfica a través del editor de formas de onda que incorpora la herramienta. Se aprecia el interés del fabricante por intentar facilitar esta labor, de definición de formas de onda, al posibilitar la definición a través de 3 métodos distintos; pero el resultado no es del todo satisfactorio ya que la definición de formas de onda, sobre todo si han de tener saltos a intervalos no regulares, sigue siendo una labor lenta y que requiere del usuario la realización de muchos pasos hasta conseguir la forma deseada. Las dificultades que presenta la definición de formas de onda, pueden provocar que los alumnos no realicen las simulaciones necesarias y que por desmotivación no vean la necesidad de realizar simulaciones, ni los beneficios de las mismas. Por estos motivos los alumnos serían reticentes a la realización de simulaciones, lo cual es contrario a lo que se pretende, ya que en un ciclo de diseño real se han de consumir mayores esfuerzos en la simulación que en el resto del diseño; ya que la realización de simulaciones exhaustivas revierte en forma de ahorro de costes, al minimizar las reprogramaciones de dispositivos.

Otra dificultad a la hora de realizar simulaciones es que el visor de formas de onda es muy básico, lo que provoca que la revisión de los resultados de las simulaciones sea una labor poco ágil. Esta desventaja unida a la anterior son razones de mucho peso que lastran a este entorno a la hora de poder ser elegido para la enseñanza de herramientas de CAD.

El entorno obliga a realizar una elección prematura del dispositivo y aunque puede cambiarse en cualquier momento, esta elección inicial ha de realizarse antes incluso de comenzar a capturar el esquema del circuito. Esta obligación puede confundir al alumno en el orden del ciclo de diseño, al elegir el dispositivo (que es físico) antes de la captura y simulación lógica del diseño. El orden ha de ser, primero la parte lógica y después la parte física.

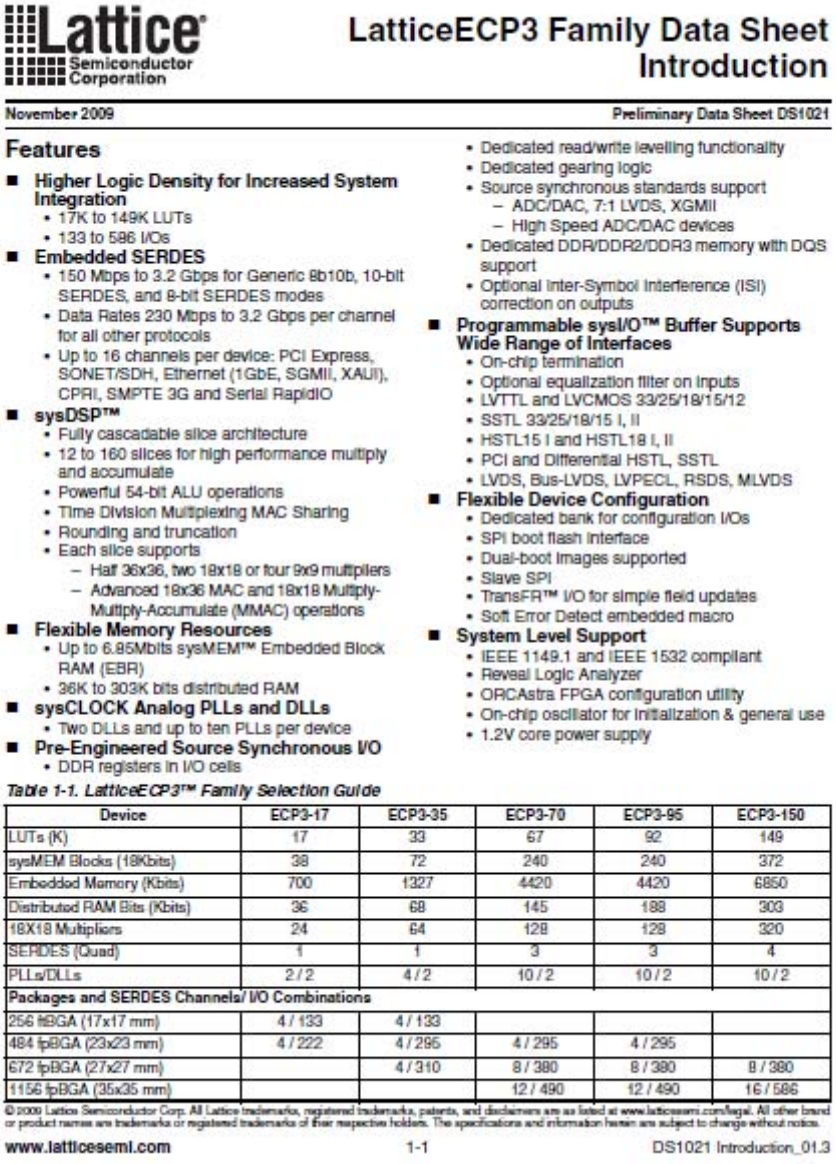
Al ser un entorno propiedad del fabricante, los únicos dispositivos físicos que pueden ser programados con él, son los del propio fabricante

El fabricante ofrece una versión de evaluación del entorno de forma gratuita, lo cual es una ventaja. Pero el hecho de que su duración sea por un tiempo limitado a un año podría convertirse en un inconveniente.

Dado que la gran mayoría de las fortalezas de éste están también presentes en el resto de entornos, como podría ser MAX+PLUS II del fabricante Altera, y estos otros entornos no presentan las graves deficiencias que tienen la definición de estímulos y la visualización de los resultados de las simulaciones en el entorno ispLEVER, unido al peso que tienen las simulaciones en los diseños llevados a cabo a nivel profesional; hace que tenga que considerar el entorno ispLEVER Classic 1.2 como poco adecuado para la enseñanza de CAD. Esto es así ya que según mi opinión, los inconvenientes que provoca a la hora de realizar simulaciones dificultan la asimilación por parte de los alumnos de parte de lo que se quiere inculcar a la hora de utilizar herramientas de CAD; como es el gran peso que tienen las simulaciones en el proceso de diseño.

V Capítulo V: “Documentación del fabricante”

V.1 DISPOSITIVOS LATTICE ECP3



Lattice Semiconductor Corporation

LatticeECP3 Family Data Sheet Introduction

November 2009 — Preliminary Data Sheet DS1021

Features

- **Higher Logic Density for Increased System Integration**
  - 17K to 149K LUTs
  - 133 to 586 I/Os
- **Embedded SERDES**
  - 150 Mbps to 3.2 Gbps for Generic 8b10b, 10-bit SERDES, and 8-bit SERDES modes
  - Data Rates 230 Mbps to 3.2 Gbps per channel for all other protocols
  - Up to 16 channels per device: PCI Express, SONET/SDH, Ethernet (1GbE, SGMII, XAUI), CPRI, SMPTE 3G and Serial RapidIO
- **sysDSP™**
  - Fully cascadable slice architecture
  - 12 to 160 slices for high performance multiply and accumulate
  - Powerful 54-bit ALU operations
  - Time Division Multiplexing MAC Sharing
  - Rounding and truncation
  - Each slice supports
    - Half 36x36, two 18x18 or four 9x9 multipliers
    - Advanced 18x36 MAC and 18x18 Multiply-Multiply-Accumulate (MMAC) operations
- **Flexible Memory Resources**
  - Up to 6.85Mbits sysMEM™ Embedded Block RAM (EBR)
  - 36K to 303K bits distributed RAM
- **sysCLOCK Analog PLLs and DLLs**
  - Two DLLs and up to ten PLLs per device
- **Pre-Engineered Source Synchronous I/O**
  - DDR registers in I/O cells
  - Dedicated read/write levelling functionality
  - Dedicated gearing logic
  - Source synchronous standards support
    - ADC/DAC, 7:1 LVDS, XGMII
    - High Speed ADC/DAC devices
  - Dedicated DDR/DDR2/DDR3 memory with DQS support
  - Optional Inter-Symbol Interference (ISI) correction on outputs
- **Programmable sysI/O™ Buffer Supports Wide Range of Interfaces**
  - On-chip termination
  - Optional equalization filter on inputs
  - LVTTL and LVCMOS 33/25/18/15/12
  - SSTL 33/25/18/15 I, II
  - HSTL15 I and HSTL18 I, II
  - PCI and Differential HSTL, SSTL
  - LVDS, Bus-LVDS, LVPECL, RSDS, MLVDS
- **Flexible Device Configuration**
  - Dedicated bank for configuration I/Os
  - SPI boot flash interface
  - Dual-boot images supported
  - Slave SPI
  - TransFR™ I/O for simple field updates
  - Soft Error Detect embedded macro
- **System Level Support**
  - IEEE 1149.1 and IEEE 1532 compliant
  - Reveal Logic Analyzer
  - ORCAstra FPGA configuration utility
  - On-chip oscillator for initialization & general use
  - 1.2V core power supply

Table 1-1. LatticeECP3™ Family Selection Guide

Device	ECP3-17	ECP3-35	ECP3-70	ECP3-95	ECP3-150
LUTs (K)	17	33	67	92	149
sysMEM Blocks (18Kbits)	38	72	240	240	372
Embedded Memory (Kbits)	700	1327	4420	4420	6850
Distributed RAM Bits (Kbits)	36	68	145	188	303
18X18 Multipliers	24	64	128	128	320
SERDES (Quad)	1	1	3	3	4
PLLs/DLLs	2 / 2	4 / 2	10 / 2	10 / 2	10 / 2
Packages and SERDES Channels/ I/O Combinations					
256 ftBGA (17x17 mm)	4 / 133	4 / 133			
484 fpBGA (23x23 mm)	4 / 222	4 / 295	4 / 295	4 / 295	
672 fpBGA (27x27 mm)		4 / 310	8 / 380	8 / 380	8 / 380
1156 fpBGA (35x35 mm)			12 / 490	12 / 490	16 / 586

Figura V.1

Introduction

The LatticeECP3™ (EConomy Plus Third generation) family of FPGA devices is optimized to deliver high performance features such as an enhanced DSP architecture, high speed SERDES and high speed source synchronous interfaces in an economical FPGA fabric. This combination is achieved through advances in device architecture and the use of 65nm technology making the devices suitable for high-volume, high-speed, low-cost applications.

The LatticeECP3 device family expands look-up-table (LUT) capacity to 149K logic elements and supports up to 486 user I/Os. The LatticeECP3 device family also offers up to 320 18x18 multipliers and a wide range of parallel I/O standards.

The LatticeECP3 FPGA fabric is optimized with high performance and low cost in mind. The LatticeECP3 devices utilize reconfigurable SRAM logic technology and provide popular building blocks such as LUT-based logic, distributed and embedded memory, Phase Locked Loops (PLLs), Delay Locked Loops (DLLs), pre-engineered source synchronous I/O support, enhanced sysDSP slices and advanced configuration support, including encryption and dual-boot capabilities.

The pre-engineered source synchronous logic implemented in the LatticeECP3 device family supports a broad range of interface standards, including DDR3, XGMII and 7:1 LVDS.

The LatticeECP3 device family also features high speed SERDES with dedicated PCS functions. High jitter tolerance and low transmit jitter allow the SERDES plus PCS blocks to be configured to support an array of popular data protocols including PCI Express, SMPTE, Ethernet (XAUI, GbE, and SGMII) and CPRI. Transmit Pre-emphasis and Receive Equalization settings make the SERDES suitable for transmission and reception over various forms of media.

The LatticeECP3 devices also provide flexible, reliable and secure configuration options, such as dual-boot capability, bit-stream encryption, and TransFR field upgrade features.

The ispLEVER® design tool suite from Lattice allows large complex designs to be efficiently implemented using the LatticeECP3 FPGA family. Synthesis library support for LatticeECP3 is available for popular logic synthesis tools. The ispLEVER tool uses the synthesis tool output along with the constraints from its floor planning tools to place and route the design in the LatticeECP3 device. The ispLEVER tool extracts the timing from the routing and back-annotates it into the design for timing verification.

Lattice provides many pre-engineered IP (Intellectual Property) ispLeverCORE™ modules for the LatticeECP3 family. By using these configurable soft core IPs as standardized blocks, designers are free to concentrate on the unique aspects of their design, increasing their productivity.

Figura V.2

V.2 DISPOSITIVOS LATTICE ECP2/M

Lattice Semiconductor Corporation

LatticeECP2/M Family Data Sheet Introduction

June 2008 — Data Sheet DS1006

Features

- **High Logic Density for System Integration**
  - 6K to 95K LUTs
  - 90 to 583 I/Os
- **Embedded SERDES (LatticeECP2M Only)**
  - Data Rates 250 Mbps to 3.125 Gbps
  - Up to 16 channels per device
    PCI Express, Ethernet (1GbE, SGMII), OBSAI, CPRI and Serial RapidIO.
- **sysDSP™ Block**
  - 3 to 42 blocks for high performance multiply and accumulate
  - Each block supports
    - One 36x36, four 18X18 or eight 9X9 multipliers
- **Flexible Memory Resources**
  - 55Kbits to 5308Kbits sysMEM™ Embedded Block RAM (EBR)
    - 18Kbit block
    - Single, pseudo dual and true dual port
    - Byte Enable Mode support
  - 12K to 202Kbits distributed RAM
    - Single port and pseudo dual port
- **sysCLOCK Analog PLLs and DLLs**
  - Two GPLLs and up to six SPLLs per device
    - Clock multiply, divide, phase & delay adjust
    - Dynamic PLL adjustment
  - Two general purpose DLLs per device
- **Pre-Engineered Source Synchronous I/O**
  - DDR registers in I/O cells
  - Dedicated gearing logic
  - Source synchronous standards support
    - SPI4.2, SFI4 (DDR Mode), XGMII
    - High Speed ADC/DAC devices
  - Dedicated DDR and DDR2 memory support
    - DDR1: 400 (200MHz) / DDR2: 533 (266MHz)
  - Dedicated DQS support
- **Programmable sysI/O™ Buffer Supports Wide Range Of Interfaces**
  - LVTTL and LVCMOS 33/25/18/15/12
  - SSTL 3/2/18 I, II
  - HSTL15 I and HSTL18 I, II
  - PCI and Differential HSTL, SSTL
  - LVDS, RSDS, Bus-LVDS, MLVDS, LVPECL
- **Flexible Device Configuration**
  - 1149.1 Boundary Scan compliant
  - Dedicated bank for configuration I/Os
  - SPI boot flash interface
  - Dual boot images supported
  - TransFR™ I/O for simple field updates
  - Soft Error Detect macro embedded
- **Optional Bitstream Encryption (LatticeECP2/M “S” Versions Only)**
- **System Level Support**
  - IspTRACY™ internal logic analyzer capability
  - On-chip oscillator for initialization & general use
  - 1.2V power supply

Table 1-1. LatticeECP2 (Including “S-Series”) Family Selection

Device	ECP2-6	ECP2-12	ECP2-20	ECP2-35	ECP2-50	ECP2-70
LUTs (K)	6	12	21	32	48	68
Distributed RAM (Kbits)	12	24	42	64	96	136
EBR SRAM (Kbits)	55	221	276	332	387	1032
EBR SRAM Blocks	3	12	15	18	21	60
sysDSP Blocks	3	6	7	8	18	22
18x18 Multipliers	12	24	28	32	72	88
GPLL + SPLL + DLL	2+0+2	2+0+2	2+0+2	2+0+2	2+2+2	2+4+2
Maximum Available I/O	190	297	402	450	500	583
Packages and I/O Combinations						
144-pin TQFP (20 x 20 mm)	90	93				
208-pin PQFP (28 x 28 mm)		131	131			
256-ball fpBGA (17 x 17 mm)	190	193	193			
484-ball fpBGA (23 x 23 mm)		297	331	331	339	
672-ball fpBGA (27 x 27 mm)			402	450	500	500
900-ball fpBGA (31 x 31 mm)						583

Figura V.3

Table 1-2. LatticeECP2M (Including “S-Series”) Family Selection

Device	ECP2M20	ECP2M35	ECP2M50	ECP2M70	ECP2M100
LUTs (K)	19	34	48	67	95
sysMEM Blocks (18kb)	66	114	225	246	288
Embedded Memory (Kbits)	1217	2101	4147	4534	5308
Distributed Memory (Kbits)	41	71	101	145	202
sysDSP Blocks	6	8	22	24	42
18x18 Multipliers	24	32	88	96	168
GPLL+SPLL+DLL	2+6+2	2+6+2	2+6+2	2+6+2	2+6+2
Maximum Available I/O	304	410	410	436	520
Packages and SERDES / I/O Combinations					
256-ball fpBGA (17 x 17 mm)	4 / 140	4 / 140			
484-ball fpBGA (23 x 23 mm)	4 / 304	4 / 303	4 / 270		
672-ball fpBGA (27 x 27 mm)		4 / 410	8 / 372		
900-ball fpBGA (31 x 31 mm)			8 / 410	16 / 416	16 / 416
1152-ball fpBGA (35 x 35 mm)				16 / 436	16 / 520

Introduction

The LatticeECP2/M family of FPGA devices is optimized to deliver high performance features such as advanced DSP blocks, high speed SERDES (LatticeECP2M family only) and high speed source synchronous interfaces in an economical FPGA fabric. This combination was achieved through advances in device architecture and the use of 90nm technology.

The LatticeECP2/M FPGA fabric is optimized with high performance and low cost in mind. The LatticeECP2/M devices include LUT-based logic, distributed and embedded memory, Phase Locked Loops (PLLs), Delay Locked Loops (DLLs), pre-engineered source synchronous I/O support, enhanced sysDSP blocks and advanced configuration support, including encryption (“S” versions only) and dual boot capabilities.

The LatticeECP2M device family features high speed SERDES with PCS. These high jitter tolerance and low transmission jitter SERDES with PCS blocks can be configured to support an array of popular data protocols including PCI Express, Ethernet (1GbE and SGMII), OBSAI and CPRI. Transmit Pre-emphasis and Receive Equalization settings make SERDES suitable for chip to chip and small form factor backplane applications.

The ispLEVER® design tool suite from Lattice allows large complex designs to be efficiently implemented using the LatticeECP2/M FPGA family. Synthesis library support for LatticeECP2/M is available for popular logic synthesis tools. The ispLEVER tool uses the synthesis tool output along with the constraints from its floor planning tools to place and route the design in the LatticeECP2/M device. The ispLEVER tool extracts the timing from the routing and back-annotates it into the design for timing verification.

Lattice provides many pre-engineered IP (Intellectual Property) ispLeverCORE™ modules for the LatticeECP2/M family. By using these IP cores as standardized blocks, designers are free to concentrate on the unique aspects of their design, increasing their productivity.

Figura V.4

V.3 DISPOSITIVOS LATTICE ECP & EC

Lattice Semiconductor Corporation

LatticeECP/EC Family Data Sheet Introduction

May 2005 — Data Sheet

Features

- **Extensive Density and Package Options**
  - 1.5K to 32.8K LUT4s
  - 65 to 496 I/Os
  - Density migration supported
- **sysDSP™ Block (LatticeECP™ Versions)**
  - High performance multiply and accumulate
  - 4 to 8 blocks
    - 4 to 8 36x36 multipliers or
    - 16 to 32 18x18 multipliers or
    - 32 to 64 9x9 multipliers
- **Embedded and Distributed Memory**
  - 18 Kbits to 498 Kbits sysMEM™ Embedded Block RAM (EBR)
  - Up to 131 Kbits distributed RAM
  - Flexible memory resources:
    - Distributed and block memory
- **Flexible I/O Buffer**
  - Programmable sysI/O™ buffer supports wide range of interfaces:
    - LVCMOS 3.3/2.5/1.8/1.5/1.2
    - LVTTL
    - SSTL 3/2 Class I, II, SSTL18 Class I
    - HSTL 18 Class I, II, III, HSTL15 Class I, III
    - PCI
    - LVDS, Bus-LVDS, LVPECL, RSDS
- **Dedicated DDR Memory Support**
  - Implements interface up to DDR400 (200MHz)
- **sysCLOCK™ PLLs**
  - Up to four analog PLLs per device
  - Clock multiply, divide and phase shifting
- **System Level Support**
  - IEEE Standard 1149.1 Boundary Scan, plus ispTRACY™ internal logic analyzer capability
  - SPI boot flash interface
  - 1.2V power supply
- **Low Cost FPGA**
  - Features optimized for mainstream applications
  - Low cost TQFP and PQFP packaging

Table 1-1. LatticeECP/EC Family Selection Guide

Device	LFEC1	LFEC3	LFEC6/ LFECP6	LFEC10/ LFECP10	LFEC15/ LFECP15	LFEC20/ LFECP20	LFEC33/ LFECP33
PFU/PFF Rows	12	16	24	32	40	44	64
PFU/PFF Columns	16	24	32	40	48	56	64
PFUs/PFFs	192	384	768	1280	1920	2464	4096
LUTs (K)	1.5	3.1	6.1	10.2	15.4	19.7	32.8
Distributed RAM (Kbits)	6	12	25	41	61	79	131
EBR SRAM (Kbits)	18	55	92	276	350	424	498
EBR SRAM Blocks	2	6	10	30	38	46	54
sysDSP Blocks[1]	—	—	4	5	6	7	8
18x18 Multipliers[1]	—	—	16	20	24	28	32
V_{CC} Voltage (V)	1.2	1.2	1.2	1.2	1.2	1.2	1.2
Number of PLLs	2	2	2	4	4	4	4
Packages and I/O Combinations:							
100-pin TQFP (14 x 14 mm)	67	67					
144-pin TQFP (20 x 20 mm)	97	97	97				
208-pin PQFP (28 x 28 mm)	112	145	147	147			
256-ball fpBGA (17 x 17 mm)		160	195	195	195		
484-ball fpBGA (23 x 23 mm)			224	288	352	360	360
672-ball fpBGA (27 x 27 mm)						400	496

1. LatticeECP devices only.

Figura V.5

Introduction

The LatticeECP/EC family of FPGA devices is optimized to deliver mainstream FPGA features at low cost. For maximum performance and value, the LatticeECP™ (EConomy Plus) FPGA concept combines an efficient FPGA fabric with high-speed dedicated functions. Lattice's first family to implement this approach is the LatticeECP-DSP™ (EConomy Plus DSP) family, providing dedicated high-performance DSP blocks on-chip. The LatticeEC™ (EConomy) family supports all the general purpose features of LatticeECP devices without dedicated function blocks to achieve lower cost solutions.

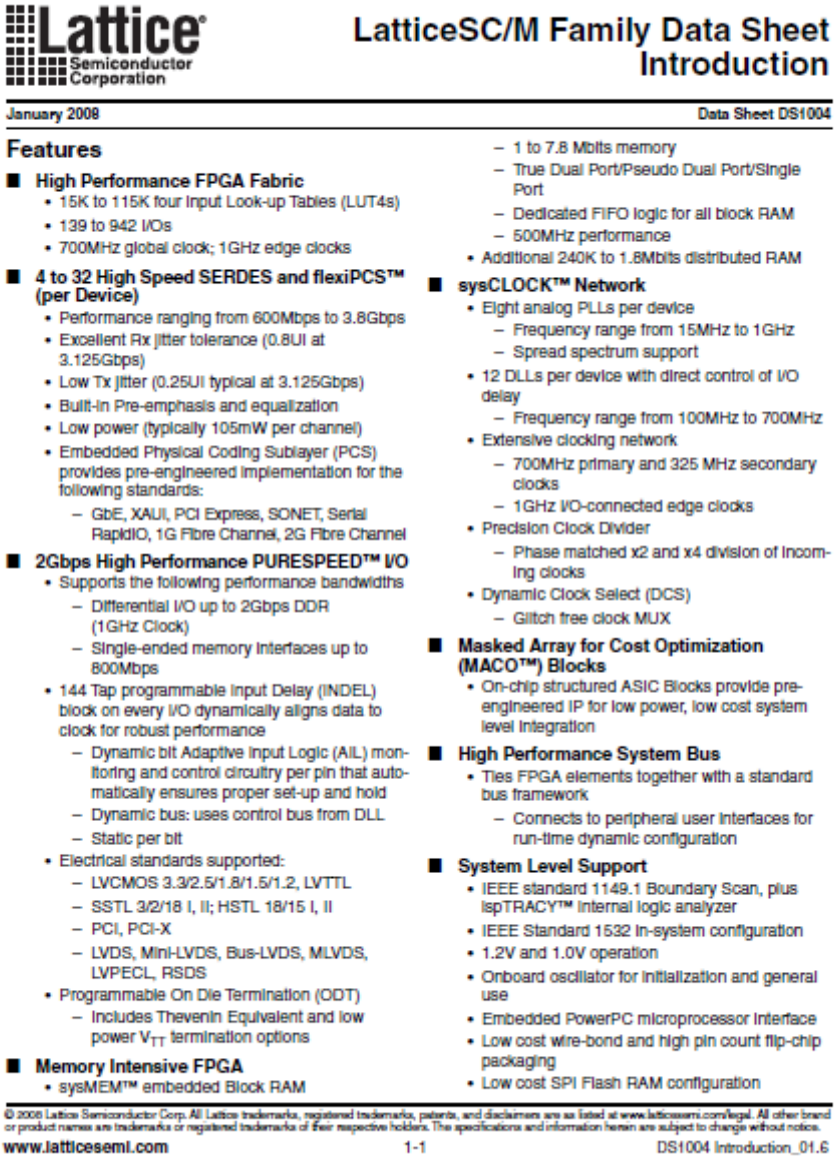
The LatticeECP/EC FPGA fabric, which was designed from the outset with low cost in mind, contains all the critical FPGA elements: LUT-based logic, distributed and embedded memory, PLLs and support for mainstream I/Os. Dedicated DDR memory interface logic is also included to support this memory that is becoming increasingly prevalent in cost-sensitive applications.

The ispLEVER® design tool suite from Lattice allows large complex designs to be efficiently implemented using the LatticeECP/EC FPGA family. Synthesis library support for LatticeECP/EC is available for popular logic synthesis tools. The ispLEVER tool uses the synthesis tool output along with the constraints from its floor planning tools to place and route the design in the LatticeECP/EC device. The ispLEVER tool extracts the timing from the routing and back-annotates it into the design for timing verification.

Lattice provides many pre-designed IP (Intellectual Property) ispLeverCORE™ modules for the LatticeECP/EC family. By using these IPs as standardized blocks, designers are free to concentrate on the unique aspects of their design, increasing their productivity.

Figura V.6

V.4 DISPOSITIVOS LATTICE SC & LATTICE SCM



Lattice Semiconductor Corporation

LatticeSC/M Family Data Sheet Introduction

January 2008 — Data Sheet DS1004

Features

High Performance FPGA Fabric
- 15K to 115K four input Look-up Tables (LUT4s)
- 139 to 942 I/Os
- 700MHz global clock; 1GHz edge clocks

4 to 32 High Speed SERDES and flexiPCS™ (per Device)
- Performance ranging from 600Mbps to 3.8Gbps
- Excellent Rx jitter tolerance (0.8UI at 3.125Gbps)
- Low Tx jitter (0.25UI typical at 3.125Gbps)
- Built-in Pre-emphasis and equalization
- Low power (typically 105mW per channel)
- Embedded Physical Coding Sublayer (PCS) provides pre-engineered implementation for the following standards:
  - GbE, XAUI, PCI Express, SONET, Serial RapidIO, 1G Fibre Channel, 2G Fibre Channel

2Gbps High Performance PURESPEED™ I/O
- Supports the following performance bandwidths
  - Differential I/O up to 2Gbps DDR (1GHz Clock)
  - Single-ended memory interfaces up to 800Mbps
- 144 Tap programmable Input Delay (INDEL) block on every I/O dynamically aligns data to clock for robust performance
  - Dynamic bit Adaptive Input Logic (AIL) monitoring and control circuitry per pin that automatically ensures proper set-up and hold
  - Dynamic bus: uses control bus from DLL
  - Static per bit
- Electrical standards supported:
  - LVCMOS 3.3/2.5/1.8/1.5/1.2, LVTTL
  - SSTL 3/2/18 I, II; HSTL 18/15 I, II
  - PCI, PCI-X
  - LVDS, Mini-LVDS, Bus-LVDS, MLVDS, LVPECL, RSDS
- Programmable On Die Termination (ODT)
  - Includes Thevenin Equivalent and low power V_{TT} termination options

Memory Intensive FPGA
- sysMEM™ embedded Block RAM
  - 1 to 7.8 Mbits memory
  - True Dual Port/Pseudo Dual Port/Single Port
  - Dedicated FIFO logic for all block RAM
  - 500MHz performance
- Additional 240K to 1.8Mbits distributed RAM

sysCLOCK™ Network
- Eight analog PLLs per device
  - Frequency range from 15MHz to 1GHz
  - Spread spectrum support
- 12 DLLs per device with direct control of I/O delay
  - Frequency range from 100MHz to 700MHz
- Extensive clocking network
  - 700MHz primary and 325 MHz secondary clocks
  - 1GHz I/O-connected edge clocks
- Precision Clock Divider
  - Phase matched x2 and x4 division of incoming clocks
- Dynamic Clock Select (DCS)
  - Glitch free clock MUX

Masked Array for Cost Optimization (MACO™) Blocks
- On-chip structured ASIC Blocks provide pre-engineered IP for low power, low cost system level integration

High Performance System Bus
- Ties FPGA elements together with a standard bus framework
  - Connects to peripheral user interfaces for run-time dynamic configuration

System Level Support
- IEEE standard 1149.1 Boundary Scan, plus ispTRACY™ internal logic analyzer
- IEEE Standard 1532 in-system configuration
- 1.2V and 1.0V operation
- Onboard oscillator for initialization and general use
- Embedded PowerPC microprocessor interface
- Low cost wire-bond and high pin count flip-chip packaging
- Low cost SPI Flash RAM configuration

Figura V.7

Table 1-1. LatticeSC Family Selection Guide

Device	SC15	SC25	SC40	SC80	SC115
LUT4s (K)	15	25	40	80	115
sysMEM Blocks (18Kb)	56	104	216	308	424
Embedded Memory (Mbits)	1.03	1.92	3.98	5.68	7.8
Max. Distributed Memory (Mbits)	0.24	0.41	0.65	1.28	1.84
Number of 3.8Gbps SERDES (Max.)	8	16	16	32	32
DLLs	12	12	12	12	12
Analog PLLs	8	8	8	8	8
MACO Blocks	4	6	10	10	12
Package I/O/SERDES Combinations (1mm ball pitch)					
256-ball fpBGA (17 x 17mm)	139/4				
900-ball fpBGA (31 x 31mm)	300/8	378/8			
1020-ball fcBGA (33 x 33mm)		476/16	562/16		
1152-ball fcBGA (35 x 35mm)			604/16	660/16	660/16
1704-ball fcBGA (42.5 x 42.5mm)				904/32	942/32

Note: The information in this preliminary data sheet is by definition not final and subject to change. Please consult the Lattice website and your local Lattice sales manager to ensure you have the latest information regarding the specifications for these products as you make critical design decisions.

The LatticeSCM devices add MACO-enabled IP functionality to the base LatticeSC devices. Table 1-2 shows the type and number of each pre-engineered IP core.

Table 1-2. LatticeSCM Family

Device	SCM15	SCM25	SCM40	SCM80	SCM115
flexiMAC Blocks • 1GbE Mode • 10GbE Mode • PCI Express Mode	1	2	2	2	4
SPI4.2 Blocks	1	2	2	2	2
Memory Controller Blocks • DDR/DDR2 DRAM Mode • QDR II/II+ SRAM Mode • RLDRAM I • RLDRAM II CIO/SIO	1	2	2	2	2
Low Speed CDR Blocks	0	0	2	2	2
PCI Express LTSSM (PHY) Blocks	1	0	2	2	2

Note: See each IP core user's guide for more information about support for specific LatticeSCM devices.

Introduction

The LatticeSC family of FPGA combines a high-performance FPGA fabric, high-speed SERDES, high-performance I/Os and large embedded RAM in a single industry leading architecture. This FPGA family is fabricated in a state of the art technology to provide one of the highest performing FPGAs in the industry.

This family of devices includes features to meet the needs of today's communication network systems. These features include SERDES with embedded advance PCS (Physical Coding sub-layer), up to 7.8 Mbits of sysMEM embedded block RAM, dedicated logic to support system level standards such as RAPIDIO, SPI4.2, SFI-4, UTOPIA, XGMII and CSIX. The devices in this family feature clock multiply, divide and phase shift PLLs, numerous DLLs and dynamic glitch free clock MUXs which are required in today's high end system designs. High speed, high bandwidth I/O make this family ideal for high throughput systems.

Figura V.8

The ispLEVER® design tool from Lattice allows large complex designs to be efficiently implemented using the LatticeSC family of FPGA devices. Synthesis library support for LatticeSC is available for popular logic synthesis tools. The ispLEVER tool uses the synthesis tool output along with the constraints from its floor planning tools to place and route the design in the LatticeSC device. The ispLEVER tool extracts the timing from the routing and back-annotates it into the design for timing verification.

Lattice provides many pre-designed IP (Intellectual Property) ispLeverCORE™ modules for the LatticeSC family. By using these IPs as standardized blocks, designers are free to concentrate on the unique aspects of their design, increasing their productivity.

Innovative high-performance FPGA architecture, high-speed SERDES with PCS support, sysMEM embedded memory and high performance I/O are combined in the LatticeSC to provide excellent performance for today's leading edge systems designs. Table 1-3 details the performance of several common functions implemented within the LatticeSC.

Table1-3. Speed Performance for Typical Functions[1]

Functions	Performance (MHz)[2]
32-bit Address Decoder	539
64-bit Address Decoder	517
32:1 Multiplexer	779
64-bit Adder (ripple)	353
32x8 Distributed Single Port (SP) RAM	768
64-bit Counter (up or down counter, non-loadable)	369
True Dual-Port 1024x18 bits	372
FIFO Port A: x36 bits, B: x9 bits	375

1. For additional information, see Typical Building BLock Function Performance table in this data sheet.
2. Advance information (-7 speed grade).

Figura V.9

V.5 DISPOSITIVOS LATTICE XP2

Lattice Semiconductor Corporation

LatticeXP2 Family Data Sheet Introduction

February 2008 — Data Sheet DS1009

Features

- **flexiFLASH™ Architecture**
  - Instant-on
  - Infinitely reconfigurable
  - Single chip
  - FlashBAK™ technology
  - Serial TAG memory
  - Design security
- **Live Update Technology**
  - TransFR™ technology
  - Secure updates with 128 bit AES encryption
  - Dual-boot with external SPI
- **sysDSP™ Block**
  - Three to eight blocks for high performance Multiply and Accumulate
  - 12 to 32 18x18 multipliers
  - Each block supports one 36x36 multiplier or four 18x18 or eight 9x9 multipliers
- **Embedded and Distributed Memory**
  - Up to 885 Kbits sysMEM™ EBR
  - Up to 83 Kbits Distributed RAM
- **sysCLOCK™ PLLs**
  - Up to four analog PLLs per device
  - Clock multiply, divide and phase shifting
- **Flexible I/O Buffer**
  - sysIO™ buffer supports:
    - LVCMOS 33/25/18/15/12; LVTTL
    - SSTL 33/25/18 class I, II
    - HSTL15 class I; HSTL18 class I, II
    - PCI
    - LVDS, Bus-LVDS, MLVDS, LVPECL, RSDS
- **Pre-engineered Source Synchronous Interfaces**
  - DDR / DDR2 interfaces up to 200 MHz
  - 7:1 LVDS interfaces support display applications
  - XGMII
- **Density And Package Options**
  - 5k to 40k LUT4s, 86 to 540 I/Os
  - csBGA, TQFP, PQFP, ftBGA and fpBGA packages
  - Density migration supported
- **Flexible Device Configuration**
  - SPI (master and slave) Boot Flash Interface
  - Dual Boot Image supported
  - Soft Error Detect (SED) macro embedded
- **System Level Support**
  - IEEE 1149.1 and IEEE 1532 Compliant
  - On-chip oscillator for initialization & general use
  - Devices operate with 1.2V power supply

Table 1-1. LatticeXP2 Family Selection Guide

Device	XP2-5	XP2-8	XP2-17	XP2-30	XP2-40
LUTs (K)	5	8	17	29	40
Distributed RAM (KBits)	10	18	35	56	83
EBR SRAM (KBits)	166	221	276	387	885
EBR SRAM Blocks	9	12	15	21	48
sysDSP Blocks	3	4	5	7	8
18 x 18 Multipliers	12	16	20	28	32
V_{CC} Voltage	1.2	1.2	1.2	1.2	1.2
GPLL	2	2	4	4	4
Max Available I/O	172	201	358	472	540
Packages and I/O Combinations					
132-Ball csBGA (8 x 8 mm)	86	86			
144-Pin TQFP (20 x 20 mm)	100	100			
208-Pin PQFP (28 x 28 mm)	146	146	146		
256-Ball ftBGA (17 x17 mm)	172	201	201	201	
484-Ball fpBGA (23 x 23 mm)			358	363	363
672-Ball fpBGA (27 x 27 mm)				472	540

Figura V.10

Introduction

LatticeXP2 devices combine a Look-up Table (LUT) based FPGA fabric with non-volatile Flash cells in an architecture referred to as flexiFLASH.

The flexiFLASH approach provides benefits including instant-on, infinite reconfigurability, on chip storage with FlashBAK embedded block memory and Serial TAG memory and design security. The parts also support Live Update technology with TransFR, 128-bit AES Encryption and Dual-boot technologies.

The LatticeXP2 FPGA fabric was optimized for the new technology from the outset with high performance and low cost in mind. LatticeXP2 devices include LUT-based logic, distributed and embedded memory, Phase Locked Loops (PLLs), pre-engineered source synchronous I/O support and enhanced sysDSP blocks.

The ispLEVER® design tool from Lattice allows large and complex designs to be efficiently implemented using the LatticeXP2 family of FPGA devices. Synthesis library support for LatticeXP2 is available for popular logic synthesis tools. The ispLEVER tool uses the synthesis tool output along with the constraints from its floor planning tools to place and route the design in the LatticeXP2 device. The ispLEVER tool extracts the timing from the routing and back-annotates it into the design for timing verification.

Lattice provides many pre-designed Intellectual Property (IP) ispLeverCORE™ modules for the LatticeXP2 family. By using these IPs as standardized blocks, designers are free to concentrate on the unique aspects of their design, increasing their productivity.

Figura V.11

V.6 DISPOSITIVOS LATTICE XP

Lattice Semiconductor Corporation

LatticeXP Family Data Sheet Introduction

July 2007 — Data Sheet DS1001

Features

- **Non-volatile, Infinitely Reconfigurable**
  - Instant-on – powers up in microseconds
  - No external configuration memory
  - Excellent design security, no bit stream to intercept
  - Reconfigure SRAM based logic in milliseconds
  - SRAM and non-volatile memory programmable through system configuration and JTAG ports
- **Sleep Mode**
  - Allows up to 1000x static current reduction
- **TransFR™ Reconfiguration (TFR)**
  - In-field logic update while system operates
- **Extensive Density and Package Options**
  - 3.1K to 19.7K LUT4s
  - 62 to 340 I/Os
  - Density migration supported
- **Embedded and Distributed Memory**
  - 54 Kbits to 396 Kbits sysMEM™ Embedded Block RAM
  - Up to 79 Kbits distributed RAM
  - Flexible memory resources:
    - Distributed and block memory
- **Flexible I/O Buffer**
  - Programmable sysIO™ buffer supports wide range of interfaces:
    - LVCMOS 3.3/2.5/1.8/1.5/1.2
    - LVTTL
    - SSTL 18 Class I
    - SSTL 3/2 Class I, II
    - HSTL15 Class I, III
    - HSTL 18 Class I, II, III
    - PCI
    - LVDS, Bus-LVDS, LVPECL, RSDS
- **Dedicated DDR Memory Support**
  - Implements interface up to DDR333 (166MHz)
- **sysCLOCK™ PLLs**
  - Up to 4 analog PLLs per device
  - Clock multiply, divide and phase shifting
- **System Level Support**
  - IEEE Standard 1149.1 Boundary Scan, plus ispTRACY™ internal logic analyzer capability
  - Onboard oscillator for configuration
  - Devices operate with 3.3V, 2.5V, 1.8V or 1.2V power supply

Table 1-1. LatticeXP Family Selection Guide

Device	LFXP3	LFXP6	LFXP10	LFXP15	LFXP20
PFU/PFF Rows	16	24	32	40	44
PFU/PFF Columns	24	30	38	48	56
PFU/PFF (Total)	384	720	1216	1932	2464
LUTs (K)	3	6	10	15	20
Distributed RAM (KBits)	12	23	39	61	79
EBR SRAM (KBits)	54	72	216	324	396
EBR SRAM Blocks	6	8	24	36	44
V_{CC} Voltage	1.2/1.8/2.5/3.3V	1.2/1.8/2.5/3.3V	1.2/1.8/2.5/3.3V	1.2/1.8/2.5/3.3V	1.2/1.8/2.5/3.3V
PLLs	2	2	4	4	4
Max. I/O	136	188	244	300	340
Packages and I/O Combinations:					
100-pin TQFP (14 x 14 mm)	62				
144-pin TQFP (20 x 20 mm)	100	100			
208-pin PQFP (28 x 28 mm)	136	142			
256-ball fpBGA (17 x 17 mm)		188	188	188	188
388-ball fpBGA (23 x 23 mm)			244	268	268
484-ball fpBGA (23 x 23 mm)				300	340

Figura V.12

Introduction

The LatticeXP family of FPGA devices combine logic gates, embedded memory and high performance I/Os in a single architecture that is both non-volatile and infinitely reconfigurable to support cost-effective system designs.

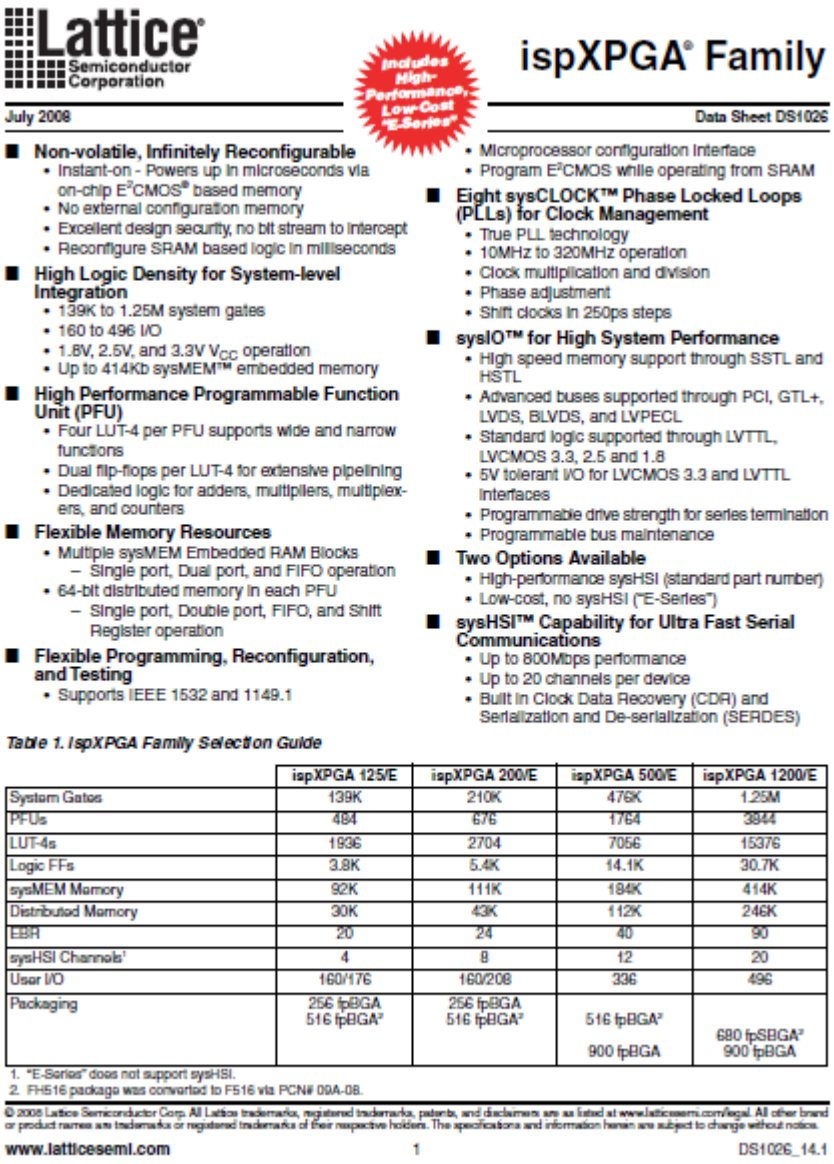
The re-programmable non-volatile technology used in the LatticeXP family is the next generation ispXP™ technology. With this technology, expensive external configuration memories are not required and designs are secured from unauthorized read-back. In addition, instant-on capability allows for easy interfacing in many applications.

The ispLEVER® design tool from Lattice allows large complex designs to be efficiently implemented using the LatticeXP family of FPGA devices. Synthesis library support for LatticeXP is available for popular logic synthesis tools. The ispLEVER tool uses the synthesis tool output along with the constraints from its floor planning tools to place and route the design in the LatticeXP device. The ispLEVER tool extracts the timing from the routing and back-annotates it into the design for timing verification.

Lattice provides many pre-designed IP (Intellectual Property) ispLeverCORE™ modules for the LatticeXP family. By using these IPs as standardized blocks, designers are free to concentrate on the unique aspects of their design, increasing their productivity.

Figura V.13

V.7 DISPOSITIVOS ISPXPGA



Lattice Semiconductor Corporation

Includes High-Performance, Low-Cost "E-Series"

ispXPGA® Family

July 2008 — Data Sheet DS1026

- **Non-volatile, Infinitely Reconfigurable**
  - Instant-on - Powers up in microseconds via on-chip E²CMOS® based memory
  - No external configuration memory
  - Excellent design security, no bit stream to intercept
  - Reconfigure SRAM based logic in milliseconds
- **High Logic Density for System-level Integration**
  - 139K to 1.25M system gates
  - 160 to 496 I/O
  - 1.8V, 2.5V, and 3.3V V_{CC} operation
  - Up to 414Kb sysMEM™ embedded memory
- **High Performance Programmable Function Unit (PFU)**
  - Four LUT-4 per PFU supports wide and narrow functions
  - Dual flip-flops per LUT-4 for extensive pipelining
  - Dedicated logic for adders, multipliers, multiplexers, and counters
- **Flexible Memory Resources**
  - Multiple sysMEM Embedded RAM Blocks
    - Single port, Dual port, and FIFO operation
  - 64-bit distributed memory in each PFU
    - Single port, Double port, FIFO, and Shift Register operation
- **Flexible Programming, Reconfiguration, and Testing**
  - Supports IEEE 1532 and 1149.1
  - Microprocessor configuration interface
  - Program E²CMOS while operating from SRAM
- **Eight sysCLOCK™ Phase Locked Loops (PLLs) for Clock Management**
  - True PLL technology
  - 10MHz to 320MHz operation
  - Clock multiplication and division
  - Phase adjustment
  - Shift clocks in 250ps steps
- **sysIO™ for High System Performance**
  - High speed memory support through SSTL and HSTL
  - Advanced buses supported through PCI, GTL+, LVDS, BLVDS, and LVPECL
  - Standard logic supported through LVTTL, LVCMOS 3.3, 2.5 and 1.8
  - 5V tolerant I/O for LVCMOS 3.3 and LVTTL interfaces
  - Programmable drive strength for series termination
  - Programmable bus maintenance
- **Two Options Available**
  - High-performance sysHSI (standard part number)
  - Low-cost, no sysHSI ("E-Series")
- **sysHSI™ Capability for Ultra Fast Serial Communications**
  - Up to 800Mbps performance
  - Up to 20 channels per device
  - Built in Clock Data Recovery (CDR) and Serialization and De-serialization (SERDES)

Table 1. ispXPGA Family Selection Guide

	ispXPGA 125/E	ispXPGA 200/E	ispXPGA 500/E	ispXPGA 1200/E
System Gates	139K	210K	476K	1.25M
PFUs	484	676	1764	3844
LUT-4s	1936	2704	7056	15376
Logic FFs	3.8K	5.4K	14.1K	30.7K
sysMEM Memory	92K	111K	184K	414K
Distributed Memory	30K	43K	112K	246K
EBR	20	24	40	90
sysHSI Channels[1]	4	8	12	20
User I/O	160/176	160/208	336	496
Packaging	256 fpBGA 516 fpBGA[2]	256 fpBGA 516 fpBGA[2]	516 fpBGA[2] 900 fpBGA	680 fpSBGA[2] 900 fpBGA

1. "E-Series" does not support sysHSI.
2. FH516 package was converted to F516 via PCN# 09A-08.

Figura V.14

ispXPGA Family Overview

The ispXPGA family of devices provides the ideal vehicle for the creation of high-performance logic designs that are both non-volatile and infinitely re-programmable. Other FPGA solutions force a compromise, being either re-programmable or non-volatile. This family couples this capability with a mainstream architecture containing the features required for today's system-level design.

The ispXPGA family is available in two options. The standard device supports sysHSI capability for ultra fast serial communications while the lower-cost "E-Series" supports the same high-performance FPGA fabric without the sysHSI Block.

Electrically Erasable CMOS (E^2CMOS) memory cells provide the ispXPGA family with non-volatile capability. These allow logic to be functional microseconds after power is applied, allowing easy interfacing in many applications. This capability also means that expensive external configuration memories are not required and that designs can be secured from unauthorized read back. Internal SRAM cells allow the device to be infinitely reconfigured if desired. Both the SRAM and E^2CMOS cells can be programmed and verified through the IEEE 1532 industry standard. Additionally, the SRAM cells can be configured and read-back through the sysCONFIG™ peripheral port.

The family spans the density and I/O range required for the majority of today's logic designs, 139K to 1.25M system gates and 160 to 496 I/O. The devices are available for operation from 1.8V, 2.5V, and 3.3V power supplies, providing easy integration into the overall system.

System-level design needs are met through the incorporation of sysMEM dual-port memory blocks, sysIO advanced I/O support, and sysCLOCK Phase Locked Loops (PLLs). High-speed serial communications are supported through multiple sysHSI blocks, which provide clock data recovery (CDR) and serialization/de-serialization (SERDES).

The ispLEVER™ design tool from Lattice allows easy implementation of designs using the ispXPGA product. Synthesis library support is available for major logic synthesis tools. The ispLEVER tool takes the output from these common synthesis packages and place and routes the design in the ispXPGA product. The tool supports floor planning and the management of other constraints within the device. The tool also provides outputs to common timing analysis tools for timing analysis.

To increase designer productivity, Lattice provides a variety of pre-designed modules referred to as IP cores for the ispXPGA product. These IP cores allow designers to concentrate on the unique portions of their design while using pre-designed blocks to implement standard functions such as bus interfaces, standard communication interfaces, and memory controllers.

Through the use of advanced technology and innovative architecture the ispXPGA FPGA devices provide designers with excellent speed performance. Although design dependent, many typical designs can run at over 150MHz. Certain designs can run at over 300MHz. Table 2 details the performance of several building blocks commonly used by logic designers.

Table 2. ispXPGA Speed Performance for Typical Building Blocks

Function	Performance
8:1 Asynch MUX	150 MHz
1:32 Asynch Demultiplexer	125 MHz
8 x 8 2-LL Pipelined Multiplier	225 MHz
32-bit Up/Down Counter	290 MHz
32-bit Shift Register	360 MHz

Figura V.15

V.8 DISPOSITIVOS MACHXO2

LATTICE SEMICONDUCTOR

MachXO2 Family Data Sheet Introduction

January 2013 — Data Sheet DS1035

Features

- **Flexible Logic Architecture**
  - Six devices with 256 to 6864 LUT4s and 19 to 335 I/Os
- **Ultra Low Power Devices**
  - Advanced 65 nm low power process
  - As low as 19 µW standby power
  - Programmable low swing differential I/Os
  - Stand-by mode and other power saving options
- **Embedded and Distributed Memory**
  - Up to 240 Kbits sysMEM™ Embedded Block RAM
  - Up to 54 Kbits Distributed RAM
  - Dedicated FIFO control logic
- **On-Chip User Flash Memory**
  - Up to 256 Kbits of User Flash Memory
  - 100,000 write cycles
  - Accessible through WISHBONE, SPI, I^2C and JTAG interfaces
  - Can be used as soft processor PROM or as Flash memory
- **Pre-Engineered Source Synchronous I/O**
  - DDR registers in I/O cells
  - Dedicated gearing logic
  - 7:1 Gearing for Display I/Os
  - Generic DDR, DDRX2, DDRX4
  - Dedicated DDR/DDR2/LPDDR memory with DQS support
- **High Performance, Flexible I/O Buffer**
  - Programmable sysIO™ buffer supports wide range of interfaces:
    - LVCMOS 3.3/2.5/1.8/1.5/1.2
    - LVTTL
    - PCI
    - LVDS, Bus-LVDS, MLVDS, RSDS, LVPECL
    - SSTL 25/18
    - HSTL 18
    - Schmitt trigger inputs, up to 0.5V hysteresis
  - I/Os support hot socketing
  - On-chip differential termination
  - Programmable pull-up or pull-down mode
- **Flexible On-Chip Clocking**
  - Eight primary clocks
  - Up to two edge clocks for high-speed I/O interfaces (top and bottom sides only)
  - Up to two analog PLLs per device with fractional-n frequency synthesis
    - Wide input frequency range (10 MHz to 400 MHz)
- **Non-volatile, Infinitely Reconfigurable**
  - Instant-on – powers up in microseconds
  - Single-chip, secure solution
  - Programmable through JTAG, SPI or I^2C
  - Supports background programming of non-volatile memory
  - Optional dual boot with external SPI memory
- **TransFR™ Reconfiguration**
  - In-field logic update while system operates
- **Enhanced System Level Support**
  - On-chip hardened functions: SPI, I^2C, timer/counter
  - On-chip oscillator with 5.5% accuracy
  - Unique TraceID for system tracking
  - One Time Programmable (OTP) mode
  - Single power supply with extended operating range
  - IEEE Standard 1149.1 boundary scan
  - IEEE 1532 compliant in-system programming
- **Broad Range of Package Options**
  - TQFP, WLCSP, ucBGA, csBGA, caBGA, ftBGA, fpBGA, QFN package options
  - Small footprint package options
    - As small as 2.5x2.5mm
  - Density migration supported
  - Advanced halogen-free packaging

Figura V.16

Table 1-1. MachXO2™ Family Selection Guide

		XO2-256	XO2-640	XO2-640U[1]	XO2-1200	XO2-1200U[1]	XO2-2000	XO2-2000U[1]	XO2-4000	XO2-7000
LUTs		256	640	640	1280	1280	2112	2112	4320	6864
Distributed RAM (Kbits)		2	5	5	10	10	16	16	34	54
EBR SRAM (Kbits)		0	18	64	64	74	74	92	92	240
Number of EBR SRAM Blocks (9 Kbits/block)		0	2	7	7	8	8	10	10	26
UFM (Kbits)		0	24	64	64	80	80	96	96	256
Device Options	HC[2]	■	■	■	■	■	■	■	■	■
	HE[3]						■	■	■	■
	ZE[4]	■	■		■		■		■	■
Number of PLLs		0	0	1	1	1	1	2	2	2
Hardened Functions: I²C		2	2	2	2	2	2	2	2	2
SPI		1	1	1	1	1	1	1	1	1
Timer/Counter		1	1	1	1	1	1	1	1	1
Packages		I/Os								
25 WLCSP[5] (2.5 x 2.5mm, 0.4mm)					18					
32 QFN[6] (5 x 5mm, 0.5mm)		21								
64 ucBGA (4 x 4mm, 0.4mm)		44								
100 TQFP (14 x 14mm)		55	78		79		79			
132 csBGA (8 x 8mm, 0.5mm)		55	79		104		104		104	
144 TQFP (20 x 20mm)				107	107		111		114	114
184 csBGA[7] (8 x 8mm, 0.5mm)									150	
256 caBGA (14 x 14mm, 0.8mm)							206		206	206
256 ftBGA (17 x 17mm, 1.0mm)						206	206		206	206
332 caBGA (17 x 17mm, 0.8mm)									274	278
484 fpBGA (23 x 23mm, 1.0mm)								278	278	334

1. Ultra high I/O device.
2. High performance with regulator – V_{CC} = 2.5V, 3.3V
3. High performance without regulator – V_{CC} = 1.2V
4. Low power without regulator – V_{CC} = 1.2V
5. WLCSP package only available for ZE devices.
6. QFN package only available for HC and ZE devices.
7. 184 csBGA package only available for HE devices.

Introduction

The MachXO2 family of ultra low power, instant-on, non-volatile PLDs has six devices with densities ranging from 256 to 6864 Look-Up Tables (LUTs). In addition to LUT-based, low-cost programmable logic these devices feature Embedded Block RAM (EBR), Distributed RAM, User Flash Memory (UFM), Phase Locked Loops (PLLs), pre-engineered source synchronous I/O support, advanced configuration support including dual-boot capability and hardened versions of commonly used functions such as SPI controller, I²C controller and timer/counter. These features allow these devices to be used in low cost, high volume consumer and system applications.

The MachXO2 devices are designed on a 65nm non-volatile low power process. The device architecture has several features such as programmable low swing differential I/Os and the ability to turn off I/O banks, on-chip PLLs

Figura V.17

and oscillators dynamically. These features help manage static and dynamic power consumption resulting in low static power for all members of the family.

The MachXO2 devices are available in two versions – ultra low power (ZE) and high performance (HC and HE) devices. The ultra low power devices are offered in three speed grades -1, -2 and -3, with -3 being the fastest. Similarly, the high-performance devices are offered in three speed grades: -4, -5 and -6, with -6 being the fastest. HC devices have an internal linear voltage regulator which supports external V_{CC} supply voltages of 3.3V or 2.5V. ZE and HE devices only accept 1.2V as the external V_{CC} supply voltage. With the exception of power supply voltage all three types of devices (ZE, HC and HE) are functionally compatible and pin compatible with each other.

The MachXO2 PLDs are available in a broad range of advanced halogen-free packages ranging from the space saving 2.5x2.5 mm WLCSP to the 23x23 mm fpBGA. MachXO2 devices support density migration within the same package. Table 1-1 shows the LUT densities, package and I/O options, along with other key parameters.

The pre-engineered source synchronous logic implemented in the MachXO2 device family supports a broad range of interface standards, including LPDDR, DDR, DDR2 and 7:1 gearing for display I/Os.

The MachXO2 devices offer enhanced I/O features such as drive strength control, slew rate control, PCI compatibility, bus-keeper latches, pull-up resistors, pull-down resistors, open drain outputs and hot socketing. Pull-up, pull-down and bus-keeper features are controllable on a "per-pin" basis.

A user-programmable internal oscillator is included in MachXO2 devices. The clock output from this oscillator may be divided by the timer/counter for use as clock input in functions such as LED control, key-board scanner and similar state machines.

The MachXO2 devices also provide flexible, reliable and secure configuration from on-chip Flash memory. These devices can also configure themselves from external SPI Flash or be configured by an external master through the JTAG test access port or through the I^2C port. Additionally, MachXO2 devices support dual-boot capability (using external Flash memory) and remote field upgrade (TransFR) capability.

Lattice provides a variety of design tools that allow complex designs to be efficiently implemented using the MachXO2 family of devices. Popular logic synthesis tools provide synthesis library support for MachXO2. Lattice design tools use the synthesis tool output along with the user-specified preferences and constraints to place and route the design in the MachXO2 device. These tools extract the timing from the routing and back-annotate it into the design for timing verification.

Lattice provides many pre-engineered IP (Intellectual Property) LatticeCORE™ modules, including a number of reference designs licensed free of charge, optimized for the MachXO2 PLD family. By using these configurable soft core IP cores as standardized blocks, users are free to concentrate on the unique aspects of their design, increasing their productivity.

Figura V.18

V.9 DISPOSITIVOS MACHXO

Lattice Semiconductor Corporation

MachXO Family Data Sheet Introduction

June 2009 — Data Sheet DS1002

Features

- **Non-volatile, Infinitely Reconfigurable**
  - Instant-on – powers up in microseconds
  - Single chip, no external configuration memory required
  - Excellent design security, no bit stream to intercept
  - Reconfigure SRAM based logic in milliseconds
  - SRAM and non-volatile memory programmable through JTAG port
  - Supports background programming of non-volatile memory
- **Sleep Mode**
  - Allows up to 100x static current reduction
- **TransFR™ Reconfiguration (TFR)**
  - In-field logic update while system operates
- **High I/O to Logic Density**
  - 256 to 2280 LUT4s
  - 73 to 271 I/Os with extensive package options
  - Density migration supported
  - Lead free/RoHS compliant packaging
- **Embedded and Distributed Memory**
  - Up to 27.6 Kbits sysMEM™ Embedded Block RAM
  - Up to 7.7 Kbits distributed RAM
  - Dedicated FIFO control logic
- **Flexible I/O Buffer**
  - Programmable sysIO™ buffer supports wide range of interfaces:
    - LVCMOS 3.3/2.5/1.8/1.5/1.2
    - LVTTL
    - PCI
    - LVDS, Bus-LVDS, LVPECL, RSDS
- **sysCLOCK™ PLLs**
  - Up to two analog PLLs per device
  - Clock multiply, divide, and phase shifting
- **System Level Support**
  - IEEE Standard 1149.1 Boundary Scan
  - Onboard oscillator
  - Devices operate with 3.3V, 2.5V, 1.8V or 1.2V power supply
  - IEEE 1532 compliant in-system programming

Introduction

The MachXO is optimized to meet the requirements of applications traditionally addressed by CPLDs and low capacity FPGAs: glue logic, bus bridging, bus interfacing, power-up control, and control logic. These devices bring together the best features of CPLD and FPGA devices on a single chip.

Table 1-1. MachXO Family Selection Guide

Device	LCMXO256	LCMXO640	LCMXO1200	LCMXO2280
LUTs	256	640	1200	2280
Dist. RAM (Kbits)	2.0	6.1	6.4	7.7
EBR SRAM (Kbits)	0	0	9.2	27.6
Number of EBR SRAM Blocks (9 Kbits)	0	0	1	3
V_{CC} Voltage	1.2/1.8/2.5/3.3V	1.2/1.8/2.5/3.3V	1.2/1.8/2.5/3.3V	1.2/1.8/2.5/3.3V
Number of PLLs	0	0	1	2
Max. I/O	78	159	211	271
Packages				
100-pin TQFP (14x14 mm)	78	74	73	73
144-pin TQFP (20x20 mm)		113	113	113
100-ball csBGA (8x8 mm)	78	74		
132-ball csBGA (8x8 mm)		101	101	101
256-ball caBGA (14x14 mm)		159	211	211
256-ball ftBGA (17x17 mm)		159	211	211
324-ball ftBGA (19x19 mm)				271

Figura V.19

The devices use look-up tables (LUTs) and embedded block memories traditionally associated with FPGAs for flexible and efficient logic implementation. Through non-volatile technology, the devices provide the single-chip, high-security, instant-on capabilities traditionally associated with CPLDs. Finally, advanced process technology and careful design will provide the high pin-to-pin performance also associated with CPLDs.

The ispLEVER® design tools from Lattice allow complex designs to be efficiently implemented using the MachXO family of devices. Popular logic synthesis tools provide synthesis library support for MachXO. The ispLEVER tools use the synthesis tool output along with the constraints from its floor planning tools to place and route the design in the MachXO device. The ispLEVER tool extracts the timing from the routing and back-annotates it into the design for timing verification.

Figura V.20

V.10 DISPOSITIVOS ISPMACH 4000ZE

Lattice Semiconductor Corporation

ispMACH® 4000ZE Family

1.8V In-System Programmable
Ultra Low Power PLDs

May 2009 — Data Sheet DS1022

Features

- **High Performance**
  - t_{MAX} = 260MHz maximum operating frequency
  - t_{PD} = 4.4ns propagation delay
  - Up to four global clock pins with programmable clock polarity control
  - Up to 80 PTs per output
- **Ease of Design**
  - Flexible CPLD macrocells with individual clock, reset, preset and clock enable controls
  - Up to four global OE controls
  - Individual local OE control per I/O pin
  - Excellent First-Time-Fit™ and refit
  - Wide input gating (36 input logic blocks) for fast counters, state machines and address decoders
- **Ultra Low Power**
  - Standby current as low as 10µA typical
  - 1.8V core; low dynamic power
  - Operational down to 1.6V V_{CC}
  - Superior solution for power sensitive consumer applications
  - **Per pin pull-up, pull-down or bus keeper control***
  - **Power Guard with multiple enable signals***
- **Broad Device Offering**
  - 32 to 256 macrocells
  - Multiple temperature range support
    - – Commercial: 0 to 90°C junction (T_J)
    - – Industrial: -40 to 105°C junction (T_J)
  - **Space-saving ucBGA and csBGA packages***
- **Easy System Integration**
  - Operation with 3.3V, 2.5V, 1.8V or 1.5V LVCMOS I/O
  - 5V tolerant I/O for LVCMOS 3.3, LVTTL, and PCI interfaces
  - Hot-socketing support
  - Open-drain output option
  - Programmable output slew rate
  - 3.3V PCI compatible
  - I/O pins with fast setup path
  - **Input hysteresis***
  - 1.8V core power supply
  - IEEE 1149.1 boundary scan testable
  - IEEE 1532 ISC compliant
  - 1.8V In-System Programmable (ISP™) using Boundary Scan Test Access Port (TAP)
  - Pb-free package options (only)
  - **On-chip user oscillator and timer***

***New enhanced features over original ispMACH 4000Z**

Table 1. ispMACH 4000ZE Family Selection Guide

	ispMACH 4032ZE	ispMACH 4064ZE	ispMACH 4128ZE	ispMACH 4256ZE
Macrocells	32	64	128	256
t_{PD} (ns)	4.4	4.7	5.8	5.8
t_S (ns)	2.2	2.5	2.9	2.9
t_{CO} (ns)	3.0	3.2	3.8	3.8
f_{MAX} (MHz)	260	241	200	200
Supply Voltages (V)	1.8V	1.8V	1.8V	1.8V
Packages[1] (I/O + Dedicated Inputs)				
48-Pin TQFP (7 x 7mm)	32+4	32+4		
64-Ball csBGA (5 x 5mm)	32+4	48+4		
64-Ball ucBGA (4 x 4mm)		48+4		
100-Pin TQFP (14 x 14mm)		64+10	64+10	64+10
132-Ball ucBGA (6 x 6mm)			96+4	
144-Pin TQFP (20 x 20mm)			96+4	96+14
144-Ball csBGA (7 x 7mm)		64+10	96+4	108+4

1. Pb-free only.

Figura V.21

Introduction

The high performance ispMACH 4000ZE family from Lattice offers an ultra low power CPLD solution. The new family is based on Lattice's industry-leading ispMACH 4000 architecture. Retaining the best of the previous generation, the ispMACH 4000ZE architecture focuses on significant innovations to combine high performance with low power in a flexible CPLD family. For example, the family's new Power Guard feature minimizes dynamic power consumption by preventing internal logic toggling due to unnecessary I/O pin activity.

The ispMACH 4000ZE combines high speed and low power with the flexibility needed for ease of design. With its robust Global Routing Pool and Output Routing Pool, this family delivers excellent First-Time-Fit, timing predictability, routing, pin-out retention and density migration.

The ispMACH 4000ZE family offers densities ranging from 32 to 256 macrocells. There are multiple density-I/O combinations in Thin Quad Flat Pack (TQFP), Chip Scale BGA (csBGA), and Ultra Chip Scale BGA (ucBGA) packages ranging from 32 to 144 pins/balls. Table 1 shows the macrocell, package and I/O options, along with other key parameters.

A user programmable internal oscillator and a timer are included in the device for tasks like LED control, keyboard scanner and similar housekeeping type state machines. This feature can be optionally disabled to save power.

The ispMACH 4000ZE family has enhanced system integration capabilities. It supports a 1.8V supply voltage and 3.3V, 2.5V, 1.8V and 1.5V interface voltages. Additionally, inputs can be safely driven up to 5.5V when an I/O bank is configured for 3.3V operation, making this family 5V tolerant. The ispMACH 4000ZE also offers enhanced I/O features such as slew rate control, PCI compatibility, bus-keeper latches, pull-up resistors, pull-down resistors, open drain outputs and hot socketing. Pull-up, pull-down and bus-keeper features are controllable on a "per-pin" basis. The ispMACH 4000ZE family members are 1.8V in-system programmable through the IEEE Standard 1532 interface. IEEE Standard 1149.1 boundary scan testing capability also allows product testing on automated test equipment. The 1532 interface signals TCK, TMS, TDI and TDO are referenced to V_{CC} (logic core).

Overview

The ispMACH 4000ZE devices consist of multiple 36-input, 16-macrocell Generic Logic Blocks (GLBs) interconnected by a Global Routing Pool (GRP). Output Routing Pools (ORPs) connect the GLBs to the I/O Blocks (IOBs), which contain multiple I/O cells. This architecture is shown in Figure 1.

Figure 1. Functional Block Diagram

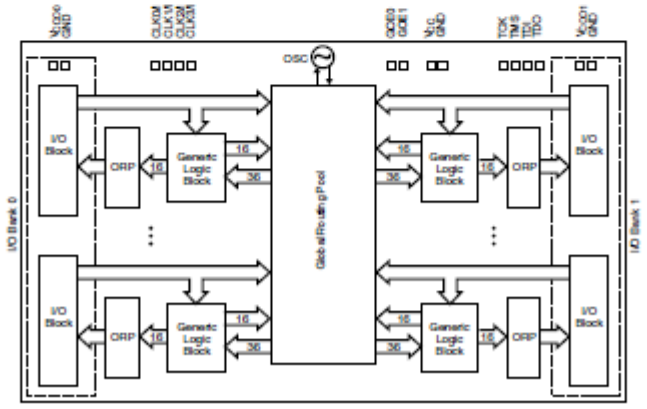


Figura V.22

V.11 DISPOSITIVOS ISPMACH 4000V/B/C/Z

Lattice Semiconductor Corporation

ispMACH® 4000V/B/C/Z Family

3.3V/2.5V/1.8V In-System Programmable SuperFAST™ High Density PLDs

May 2009 — Coolest Power, Coolest Price™ — Data Sheet DS1020

Features

High Performance
- t_{MAX} = 400MHz maximum operating frequency
- t_{PD} = 2.5ns propagation delay
- Up to four global clock pins with programmable clock polarity control
- Up to 80 PTs per output

Ease of Design
- Enhanced macrocells with individual clock, reset, preset and clock enable controls
- Up to four global OE controls
- Individual local OE control per I/O pin
- Excellent First-Time-Fit™ and refit
- Fast path, SpeedLocking™ Path, and wide-PT path
- Wide input gating (36 input logic blocks) for fast counters, state machines and address decoders

Zero Power (ispMACH 4000Z) and Low Power (ispMACH 4000V/B/C)
- Typical static current 10µA (4032Z)
- Typical static current 1.3mA (4000C)
- 1.8V core low dynamic power
- ispMACH 4000Z operational down to 1.6V V_{CC}

Broad Device Offering
- Multiple temperature range support
  - Commercial: 0 to 90°C junction (T_J)
  - Industrial: -40 to 105°C junction (T_J)
  - Extended: -40 to 130°C junction (T_J)
- For AEC-Q100 compliant devices, refer to LA-ispMACH 4000V/Z Automotive Data Sheet

Easy System Integration
- Superior solution for power sensitive consumer applications
- Operation with 3.3V, 2.5V or 1.8V LVCMOS I/O
- Operation with 3.3V (4000V), 2.5V (4000B) or 1.8V (4000C/Z) supplies
- 5V tolerant I/O for LVCMOS 3.3, LVTTL, and PCI interfaces
- Hot-socketing
- Open-drain capability
- Input pull-up, pull-down or bus-keeper
- Programmable output slew rate
- 3.3V PCI compatible
- IEEE 1149.1 boundary scan testable
- 3.3V/2.5V/1.8V In-System Programmable (ISP™) using IEEE 1532 compliant interface
- I/O pins with fast setup path
- Lead-free package options

Table 1. ispMACH 4000V/B/C Family Selection Guide

	ispMACH 4032V/B/C	ispMACH 4064V/B/C	ispMACH 4128V/B/C	ispMACH 4256V/B/C	ispMACH 4384V/B/C	ispMACH 4512V/B/C
Macrocells	32	64	128	256	384	512
I/O + Dedicated Inputs	30+2/32+4	30+2/32+4/ 64+10	64+10/92+4/ 96+4	64+10/96+14/ 128+4/160+4	128+4/192+4	128+4/208+4
t_{PD} (ns)	2.5	2.5	2.7	3.0	3.5	3.5
t_S (ns)	1.8	1.8	1.8	2.0	2.0	2.0
t_{CO} (ns)	2.2	2.2	2.7	2.7	2.7	2.7
f_{MAX} (MHz)	400	400	333	322	322	322
Supply Voltages (V)	3.3/2.5/1.8V	3.3/2.5/1.8V	3.3/2.5/1.8V	3.3/2.5/1.8V	3.3/2.5/1.8V	3.3/2.5/1.8V
Pins/Package	44 TQFP 48 TQFP	44 TQFP 48 TQFP 100 TQFP	100 TQFP 128 TQFP 144 TQFP[1]	100 TQFP 144 TQFP[1] 176 TQFP 256 fBGA[2]/ fpBGA[2,3]	176 TQFP 256 fBGA/ fpBGA[3]	176 TQFP 256 fBGA/ fpBGA[3]

1. 3.3V (4000V) only.
2. 128-I/O and 160-I/O configurations.
3. Use 256 fBGA package for all new designs. Refer to PCN#14A-07 for 256 fpBGA package discontinuance.

Figura V.23

Table 2. ispMACH 4000Z Family Selection Guide

	ispMACH 4032ZC	ispMACH 4064ZC	ispMACH 4128ZC	ispMACH 4256ZC
Macrocells	32	64	128	256
I/O + Dedicated Inputs	32+4/32+4	32+4/32+12/ 64+10/64+10	64+10/96+4	64+10/96+6/ 128+4
t_{PD} (ns)	3.5	3.7	4.2	4.5
t_S (ns)	2.2	2.5	2.7	2.9
t_{CO} (ns)	3.0	3.2	3.5	3.8
f_{MAX} (MHz)	267	250	220	200
Supply Voltage (V)	1.8	1.8	1.8	1.8
Max. Standby Icc (µA)	20	25	35	55
Pins/Package	48 TQFP 56 csBGA	48 TQFP 56 csBGA 100 TQFP 132 csBGA	100 TQFP 132csBGA	100 TQFP 132 csBGA 176 TQFP

ispMACH 4000 Introduction

The high performance ispMACH 4000 family from Lattice offers a SuperFAST CPLD solution. The family is a blend of Lattice's two most popular architectures: the ispLSI® 2000 and ispMACH 4A. Retaining the best of both families, the ispMACH 4000 architecture focuses on significant innovations to combine the highest performance with low power in a flexible CPLD family.

The ispMACH 4000 combines high speed and low power with the flexibility needed for ease of design. With its robust Global Routing Pool and Output Routing Pool, this family delivers excellent First-Time-Fit, timing predictability, routing, pin-out retention and density migration.

The ispMACH 4000 family offers densities ranging from 32 to 512 macrocells. There are multiple density-I/O combinations in Thin Quad Flat Pack (TQFP), Chip Scale BGA (csBGA) and Fine Pitch Thin BGA (ftBGA) packages ranging from 44 to 256 pins/balls. Table 1 shows the macrocell, package and I/O options, along with other key parameters.

The ispMACH 4000 family has enhanced system integration capabilities. It supports 3.3V (4000V), 2.5V (4000B) and 1.8V (4000C/Z) supply voltages and 3.3V, 2.5V and 1.8V interface voltages. Additionally, inputs can be safely driven up to 5.5V when an I/O bank is configured for 3.3V operation, making this family 5V tolerant. The ispMACH 4000 also offers enhanced I/O features such as slew rate control, PCI compatibility, bus-keeper latches, pull-up resistors, pull-down resistors, open drain outputs and hot socketing. The ispMACH 4000 family members are 3.3V/ 2.5V/1.8V In-system programmable through the IEEE Standard 1532 interface. IEEE Standard 1149.1 boundary scan testing capability also allows product testing on automated test equipment. The 1532 interface signals TCK, TMS, TDI and TDO are referenced to V_{CC} (logic core).

Overview

The ispMACH 4000 devices consist of multiple 36-input, 16-macrocell Generic Logic Blocks (GLBs) interconnected by a Global Routing Pool (GRP). Output Routing Pools (ORPs) connect the GLBs to the I/O Blocks (IOBs), which contain multiple I/O cells. This architecture is shown in Figure 1.

Figura V.24